Incident Response Planning Using a Lightweight Large Language Model with Reduced Hallucination

Kim Hammar†, Tansu Alpcan†, and Emil C. Lupu‡
† Department of Electrical and Electronic Engineering, University of Melbourne, Australia
‡ Department of Computing, Imperial College London, United Kingdom
Email: {kim.hammar,tansu.alpcan}@unimelb.edu.au, and e.c.lupu@imperial.ac.uk

***Abstract*—Timely and effective incident response is key to managing the growing frequency of cyberattacks. However, identifying the right response actions for complex systems is a major technical challenge. A promising approach to mitigate this challenge is to use the security knowledge embedded in large language models (LLMs) to assist security operators during incident handling. Recent research has demonstrated the potential of this approach, but current methods are mainly based on prompt engineering of frontier LLMs, which is costly and prone to hallucinations. We address these limitations by presenting a novel way to use an LLM for incident response planning with reduced hallucination. Our method includes three steps: fine-tuning, information retrieval, and lookahead planning. We prove that our method generates response plans with a bounded probability of hallucination and that this probability can be made arbitrarily small at the expense of increased planning time under certain assumptions. Moreover, we show that our method is lightweight and can run on commodity hardware. We evaluate our method on logs from incidents reported in the literature. The experimental results show that our method a) achieves up to 22% shorter recovery times than frontier LLMs and b) generalizes to a broad range of incident types and response actions.**

I. Introduction

Incident response refers to the coordinated actions taken to contain, mitigate, and recover from cyberattacks. Today, incident response is largely a manual process carried out by security operators [1]. While this approach can be effective, it is often slow, labor-intensive, and requires significant skills. For example, a recent study reports that organizations take an average of 73 days to respond and recover from an incident [2]. Reducing this delay requires better decision-support tools to assist operators during incident handling. Currently, the standard approach to assisting operators relies on *response playbooks* [3], which comprise predefined rules for handling specific incidents. However, playbooks still rely on security experts for configuration and are therefore difficult to keep aligned with evolving threats and system architectures [4].

To overcome these limitations, an emerging direction of research is to leverage the security knowledge encoded in large language models (LLMs) to generate effective response actions [6]–[12]. These actions can then be used as suggestions to security operators. Although this approach remains largely confined to academic settings for now, it is beginning to see commercial adoption, as exemplified by IBM's recent launch of an LLM-based response service [13]. Most of the LLM-based methods proposed in the literature so far are based on prompt engineering of frontier LLMs, such as OPENAI O3 [14]. While this approach has shown promise, it is costly and relies on an external LLM provider (e.g., GOOGLE or OPENAI), which limits flexibility. Another important concern with this approach is that frontier LLMs are not specialized for incident response, which makes them particularly prone to *hallucinations* [15], i.e., they may generate response actions that appear plausible but are incorrect or unrelated to the incident.

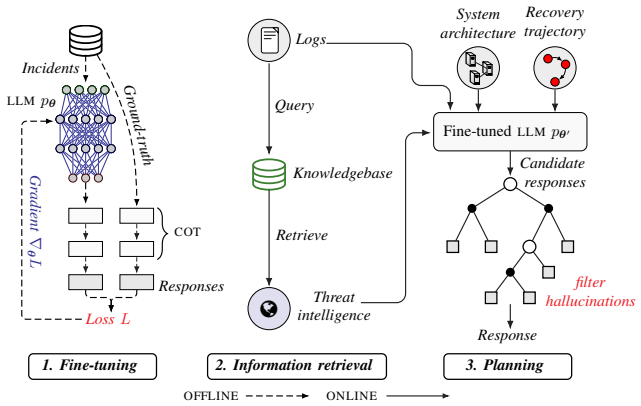


Fig. 1: The three steps of our method for incident response planning: 1. fine-tuning of a (lightweight) large language model (LLM); 2. retrieval of relevant threat intelligence; and 3. decision-theoretic planning and chain-of-thought (COT [5]) reasoning to select effective responses and filter hallucinations.

In this paper, we present a novel method that addresses these limitations and provides a principled way to use an LLM as decision support for incident response; see Fig. 1. Our method includes three main steps: (*i*) instruction fine-tuning of a lightweight LLM to align it with the phases and objectives of incident response; (*ii*) retrieval-augmented generation (RAG) to ground the LLM in current threat information and system knowledge; and (*iii*) decision-theoretic planning and chain-of-thought (COT) reasoning to generate effective response actions.

We evaluate our method based on log data from incidents reported in the literature. The results show that our method surpasses the performance of frontier LLMs (e.g., GEMINI 2.5 [16], [17]) by up to 22% while being far less resource-intensive. Moreover, we show that our method performs comparably to the PPO reinforcement learning method [18], despite not relying on incident-specific training like PPO does. We also present an ablation study assessing the contribution of the individual steps of our method. We show that all steps contribute to its performance, with fine-tuning and planning

having the greatest impact. In addition to the empirical results, we present a theoretical analysis that establishes a probabilistic upper bound on the hallucination probability of our method.

Our contributions can be summarized as follows:

- We develop a novel method for incident response that integrates a lightweight LLM with instruction fine-tuning, information retrieval, and decision-theoretic planning.
- We derive a probabilistic upper bound on the hallucination probability of our method. Under certain assumptions, this bound can be made arbitrarily small at the expense of increased planning time.
- We evaluate our method on logs from incidents reported in the literature. The results show that our method a) achieves up to 22% shorter recovery times than frontier LLMs; b) generalizes to a broad range of incidents and responses; and c) performs comparably to a reinforcement learning method that is pretrained for each incident.
- We release the first LLM fine-tuned for incident response, together with a dataset of 68,000 incidents and the corresponding responses. We also provide source code and a video demonstration of a decision-support system for incident response that implements our method [19].

II. Related Work

Since the early 2000s, there has been broad interest in developing systems that can assist security operators during incident response [20], [21]. Traditional decision-support systems are based on playbooks that map incident scenarios to sequences of response actions [20], [22], such as those provided by SPLUNK [23], CISA [24], and OASIS [25]. Although playbooks can be effective, they rely on security experts for configuration. As a consequence, they are difficult to keep up-to-date with evolving security threats and system architectures [4]. Another common critique of playbooks is that they consist of generic response actions that are difficult for non-experts to interpret and execute effectively [3]. Several research efforts have aimed to address these limitations by *automating* the generation of effective incident response strategies and functions. Four predominant approaches to such automation have emerged: decision-theoretic [26], reinforcement learning [27], game-theoretic [28]–[30], and LLM-based approaches [6].

The first three approaches share a common requirement: they need a perfect simulator (model) that captures how the system evolves in response to attacks and defensive actions. The simulator enables the computation of optimal response strategies (according to the model) through numerical optimization techniques. For example, a standard benchmark in this line of research is CAGE-2 [31], which simulates an advanced persistent threat on an enterprise network. State-of-the-art methods evaluated on this benchmark include dynamic programming [32], reinforcement learning [33], and tree search [34], all of which rely on a simulator. While these approaches can be effective when high-fidelity simulators are available, such simulators are rarely available in practice. Furthermore, the resulting response strategies are limited in scope as they are trained on a narrow set of attack vectors and response options. For instance, the CAGE-2 simulation is limited to around 20 attacker actions and defensive countermeasures [34].

A promising approach to address this drawback is to use large language models (LLMs) to automatically generate effective response actions based on system logs. This approach is not limited to a predefined set of actions and eliminates the need for a simulator. Early studies in this direction include [6]–[12], and [13]. Notably, the work in [13] is a commercial product by IBM. While these works report encouraging results, they have three key limitations: they do not provide a theoretical analysis, they do not address the risk of hallucinations, and most of them require API access to frontier LLMs.

Our method differs from prior work in several ways. It does not rely on a simulator or a manually-designed playbook, is lightweight enough to run on commodity hardware, has reduced hallucination, is accompanied by a theoretical analysis, and combines fine-tuning with retrieval-augmented generation (RAG); see Table 1. Moreover, ours is the only LLM-based method that is fully open-source (code, weights, and data).

Method	*Theory*	RAG	*Fine-tuning*	*Lightweight*	LLM	*Req. simulator*	*Manual*
OURS (Fig. 1)	✓	✓	✓	✓	✓	✗	✗
[6],[7]–[11]	✗	✗	✗	✗	✓	✗	✗
[13]	✗	?	?	?	✓	✗	✗
[12]	✗	✓	✗	✗	✓	✗	✗
[32], [34], [35]	✓	✗	✗	✓	✗	✓	✗
[33],[27], [36]–[38]	✗	✗	✗	✓	✗	✓	✗
[22], [39], [40]	✗	✗	✗	✓	✗	✗	✓

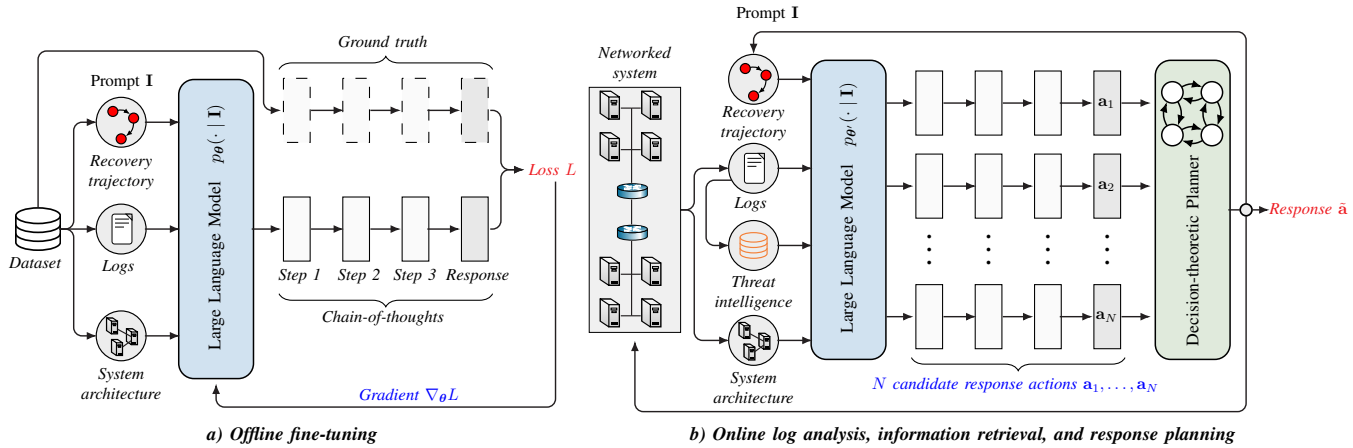
TABLE 1: Comparison between our method and related approaches, which can be grouped into three categories: those relying on playbooks (white row), those relying on a simulator for numerical optimization (red rows), and those using LLMs (blue rows). Compared to other LLM-based approaches, our method (green row) is the only method that does not depend on frontier LLMs, is lightweight enough to run on commodity hardware, has reduced hallucination probability, and is accompanied by a theoretical analysis.

Lastly, we note that a growing body of research applies LLMs to security use cases other than incident response, such as penetration testing [41], [42], security assistants [43], scanning [44], [45], threat hunting [46], verification [47], piracy [48], detection [49], fuzzing [50], [51], API design [52], network operations [53], threat intelligence [54], and decompilation [55]. Compared to these works, the main novelty of our method lies in its approach to reducing hallucinations.

III. The Incident Response Problem

Incident response involves selecting a sequence of actions that restores a networked system to a secure and operational state after a cyberattack. These actions should analyze the scope of the attack, secure forensic evidence, contain and evict the attacker, harden the system to prevent recurrence, and restore critical services. Examples of response actions include redirecting network flows, updating access control policies, patching vulnerabilities, shutting down compromised systems, and restarting operational services. From a security engineering perspective [56], incident response fits within the broader cyber resilience framework by operationalizing the response and recovery phase after a cyberattack [57].

Figure 3 illustrates the phases of incident response. Following the attack is a *response time* interval, which represents the delay between the attack and the first response. This phase is followed by a *recovery time* interval, during which response actions are deployed. When selecting these actions, the goal



a) Offline fine-tuning *b) Online log analysis, information retrieval, and response planning*

Fig. 2: The two phases of our method. In the first phase [cf. **a**)], an LLM is fine-tuned offline on a dataset of logs from 68,000 incidents paired with response plans and chain-of-thought reasoning steps [5]. In the second phase [cf. **b**)], system logs and threat intelligence are processed online by the fine-tuned LLM and used to generate N candidate responses. These responses are then evaluated via a planning procedure, which selects the most effective response.

is to restore the system to a secure and operational state as quickly as possible while minimizing operational costs. A key challenge to achieving this goal is that the information about the attack is often limited to partial indicators of compromise (e.g., log files and alerts), while the full scope and severity of the attack are unknown [58]. Another major difficulty is that even short delays in initiating the response can lead to significant costs. For example, in the event of a ransomware attack, a delay of just a few minutes may allow the malware to encrypt systems or spread laterally across the network [59].

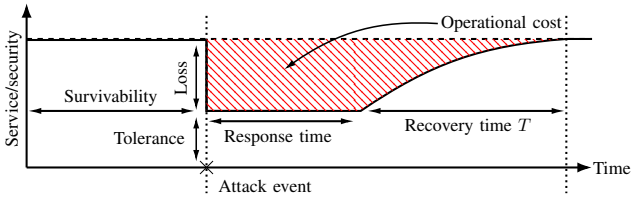


Fig. 3: Phases and performance metrics of the incident response problem.

IV. Our Method for Incident Response Planning

Motivated by the challenges described above, we develop a method for using an LLM as decision support during incident handling, i.e., to help security operators identify and execute effective response actions quickly. Broadly speaking, our method takes as input a description of an incident (e.g., system logs, security alerts, and threat intelligence) and produces as output a sequence of recommended response actions. The main challenge in generating such recommendations is to ensure that the response actions are effective despite the possibility that the LLM hallucinates. In the following subsections, we describe the steps we take to address this challenge.

A. Overview of Our Approach and System Architecture

Our method consists of three main steps: (*i*) supervised fine-tuning of a lightweight LLM to align it with the objectives of incident response; (*ii*) retrieval-augmented generation (RAG) to ground the LLM in current threat information and system knowledge; and (*iii*) decision-theoretic planning to synthesize effective response actions. These steps can be divided into two phases: an offline phase for fine-tuning and an online phase for information retrieval and response generation; see Fig. 2.

The first step of our method is to fine-tune a lightweight LLM for incident response. We conduct this fine-tuning by training the LLM on a labeled dataset of incident logs paired with corresponding response actions and reasoning steps. This training enables the LLM to learn typical patterns of incident handling. For example, it learns the logical dependencies between different phases of the response process, such as containment and eviction. Another benefit of fine-tuning is that it can reduce hallucinations; see e.g., [60].

Remark 1. *We call an* LLM lightweight *if it has significantly fewer parameters than a typical frontier* LLM. *For the experimental results reported in this paper, we use the* DEEPSEEK-R1-14B LLM, *which has* 14 *billion parameters. This parameter count is small in comparison with that of the frontier* LLM DEEPSEEK-R1, *which has* 671 *billion parameters [61].*

Once fine-tuned, the LLM can provide decision support for incident response by generating a sequence of recommended response actions when prompted with details about an incident. However, because the LLM is trained on historical incident data, it cannot generate response actions that relate to newly discovered vulnerabilities or attack techniques. To address this limitation, we augment the system logs with additional threat information retrieved online. Specifically, we automatically extract *indicators of compromise* from the logs (e.g., hostnames and vulnerability identifiers) and use them to retrieve relevant information from external sources, such as threat intelligence APIs and vulnerability databases. We then append this information to the logs before prompting the LLM. In addition to improving the quality of the response, several empirical studies have shown that such *retrieval-augmented generation* also reduces the probability of hallucinations [62].

Lastly, instead of directly selecting the response action generated by the fine-tuned LLM, we use the LLM to generate several candidate actions and select the one that is least likely to be hallucinated. In particular, we evaluate each candidate

action by using the LLM to simulate possible outcomes of the action, after which we select the action that leads to the shortest expected recovery time. This lookahead planning enforces a form of *self-consistency* [63], where actions are validated against the LLM's predicted outcomes. Such validation has been shown in prior work to reduce hallucinations; see e.g., [64], [65], and [66]. We provide a theoretical justification for why this procedure can reduce hallucination in §V.

Each of these three steps (fine-tuning, information retrieval, and planning) is detailed below, starting with fine-tuning.

B. Instruction Fine-Tuning

Our goal with fine-tuning is to make the pre-trained LLM generate appropriate response actions when prompted with system logs describing an incident. In this context, we view the pre-trained LLM as a probabilistic model that takes as input a sequence of tokens $\mathbf{x} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ and predicts the probability distribution over the subsequent token as

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{n+1} \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n), \tag{1}$$

where $\boldsymbol{\theta}$ denotes the model parameters.

The next-token prediction in (1) allows us to generate response actions as follows. We start by concatenating a description of the incident (e.g., system logs) with an instruction to generate a response action. We then pass the resulting text through a tokenizer that converts it into a sequence of tokens $\mathbf{x} = \mathbf{x}_1, \ldots, \mathbf{x}_n$. Next, we feed these tokens into the LLM to generate the next token by sampling from (1). Subsequently, we append the generated token to the prompt and feed the entire sequence back into the LLM to predict the next token. We repeat this process autoregressively until the LLM generates a special end-of-sequence token, which is produced when the LLM determines that the response action is complete, i.e., when the action has been fully specified.

Remark 2. *We place no restrictions on the form of a response action. It may be a single command, a compound procedure, or any other textual description, depending on the incident.*

To steer the LLM toward generating effective responses, we fine-tune it using supervised learning on a dataset of $68,000$ 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 a task for the LLM to perform. The associated answer $\mathbf{y}^i$ describes the correct steps to complete the task, paired with a sequence of chain-of-thought (COT [5]) reasoning steps that explain the answer. We use two types of instructions: *action-generation* instructions and *state-prediction* instructions.

In the first case, the vector $\mathbf{x}^i$ represents an instruction to generate a response to an incident. In the latter case, $\mathbf{x}^i$ represents an instruction to assess the current status of the incident response process. For example, the instruction may be to determine whether the attack has been contained, whether the system has been hardened to prevent recurrence, or whether forensic evidence has been secured. See Appendix D for the prompt templates and details on how we construct the dataset.

Remark 3. *We do not fine-tune the* LLM *for a specific incident scenario. Rather, the training dataset spans a diverse set of incidents, log formats, and response types, enabling the* LLM *to generalize across a wide range of incident scenarios.*

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

$$L = -\frac{1}{M} \sum_{i=1}^{M} \sum_{k=1}^{m_i} \ln p_{\boldsymbol{\theta}} \left(\mathbf{y}_k^i \mid \mathbf{x}^i, \mathbf{y}_1^i, \ldots, \mathbf{y}_{k-1}^i \right), \tag{2}$$

where m_i is the length of the vector $\mathbf{y}^i$. We denote the fine-tuned parameter vector by $\boldsymbol{\theta}'$ to distinguish it from $\boldsymbol{\theta}$.

Figure 4 displays the training loss curves when fine-tuning the DEEPSEEK-R1-14B LLM [61]. We run the experiment on $4\times$RTX 8000 GPUs and compare a higher learning rate (blue) with a lower one (red). We observe that the higher learning rate results in convergence to a lower loss. Additional experimental details and hyperparameters can be found in Appendix C.

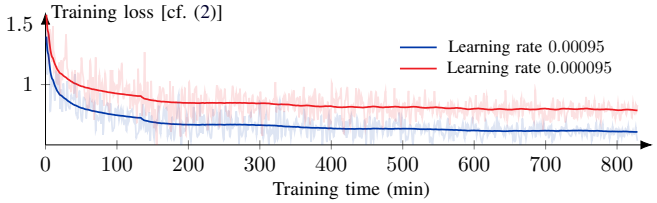


Fig. 4: Loss curves when fine-tuning the DEEPSEEK-R1-14B [61] LLM under two different learning rates. The solid lines indicate the mean loss and the shaded lines represent the loss on specific batches of training examples; cf. (2).

C. Retrieval-Augmented Response Generation (RAG)

While the fine-tuned LLM can generate effective response actions, its outputs depend on the distribution of incidents seen during training. This presents a limitation as the LLM is trained on historical data that may not reflect the most recent threat landscape. To address this challenge, we use indicators of compromise (e.g., vulnerability identifiers or hostnames) in the system logs to retrieve relevant threat intelligence from external sources. By incorporating such information at the time of action generation, the LLM can adapt its responses to reflect up-to-date threat information and system knowledge [67].

As an example, consider a scenario where the LLM is trained on data available only up to 2020. Suppose that the LLM is prompted with information about an incident that relates to a vulnerability discovered after 2020, e.g., CVE-2021-44228 [68]. In this case, the LLM may not have sufficient information to generate effective response actions, as illustrated below.

- WITHOUT RAG. Prompted only with the logs, the LLM generates the action: "*isolate host*" as it has no knowledge about the nature of the vulnerability CVE-2021-44228.
- WITH RAG. The system retrieves information about specific mitigations for CVE-2021-44228. When provided with this information, the LLM generates a response action with targeted mitigations for CVE-2021-44228, thereby reducing the time to recover from the incident.

D. Incident Response Planning

Having fine-tuned the LLM to produce response actions from incident logs, we now address the challenge of selecting the most effective action. Although the LLM can produce effective actions in many cases, it may also hallucinate and generate ineffective actions. To reduce the risk of such hallucinations, our method includes a planning procedure where we use the LLM to generate multiple candidate actions and then select the action least likely to be hallucinated, as described below.

System model. We formulate incident response planning as a stochastic shortest path problem. In this formulation, the response process evolves over a sequence of *time steps* $t = 0, 1, \ldots, T$ and the goal is to generate a sequence of *actions* $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{T-1}$ that quickly recovers the system from the incident. In other words, the goal is to minimize the *recovery time* T. To formalize this goal, we model the progress toward system recovery with a *recovery state*. We define this state based on the MITRE D3FEND [69] taxonomy as follows.

Definition 1 (Recovery state)**.** *The recovery state is a vector*

$$\mathbf{s}_t = (s_t^{\mathrm{I}}, s_t^{\mathrm{S}}, s_t^{\mathrm{F}}, s_t^{\mathrm{E}}, s_t^{\mathrm{H}}, s_t^{\mathrm{R}}), \tag{3}$$

where each component is a binary variable indicating whether a specific stage of response is completed. In particular,

- ***Containment****:* $s_t^{\mathrm{I}} = 1$ *if the attack has been isolated and stopped from spreading;* $s_t^{\mathrm{I}} = 0$ *otherwise.*
- ***Assessment****:* $s_t^{\mathrm{S}} = 1$ *if the scope and severity of the attack have been determined;* $s_t^{\mathrm{S}} = 0$ *otherwise.*
- ***Preservation****:* $s_t^{\mathrm{F}} = 1$ *if forensic evidence related to the incident has been preserved;* $s_t^{\mathrm{F}} = 0$ *otherwise.*
- ***Eviction****:* $s_t^{\mathrm{E}} = 1$ *if the attacker's access has been revoked and potential malicious code or processes have been removed from the system;* $s_t^{\mathrm{E}} = 0$ *otherwise.*
- ***Hardening****:* $s_t^{\mathrm{H}} = 1$ *if the system has been hardened to prevent recurrence of the same attack;* $s_t^{\mathrm{H}} = 0$ *otherwise.*
- ***Restoration****:* $s_t^{\mathrm{R}} = 1$ *if services have been restarted and user access has been restored;* $s_t^{\mathrm{R}} = 0$ *otherwise.*

Given this definition of the recovery state, we define the *recovery time* to be the number of time steps until the *terminal recovery state* $\mathbf{s} = (1, 1, 1, 1, 1, 1)$ is reached. Since both the system and the attacker may behave stochastically, we model the recovery time as a random variable, as defined below.

Definition 2 (Recovery time)**.** *The recovery time* T *is a random variable that takes on values in the set* $\{1, 2, \ldots\}$ *and represents the time to reach the terminal state, i.e.,*

$$T = \inf\{t \mid t > 0, \mathbf{s}_t = (1, 1, 1, 1, 1, 1)\}.$$

To illustrate the preceding definitions, we show two possible state trajectories $\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_T$ in Fig. 5. As shown in the figure, several response actions may achieve the same effect on the recovery state. For example, the severity of the attack can be determined in several ways. Moreover, certain response actions can lead to shorter recovery times by skipping intermediate steps. For instance, in the event of a denial of service (DOS) attack, containment and eviction can often be achieved simultaneously by appropriate filtering of the network traffic.

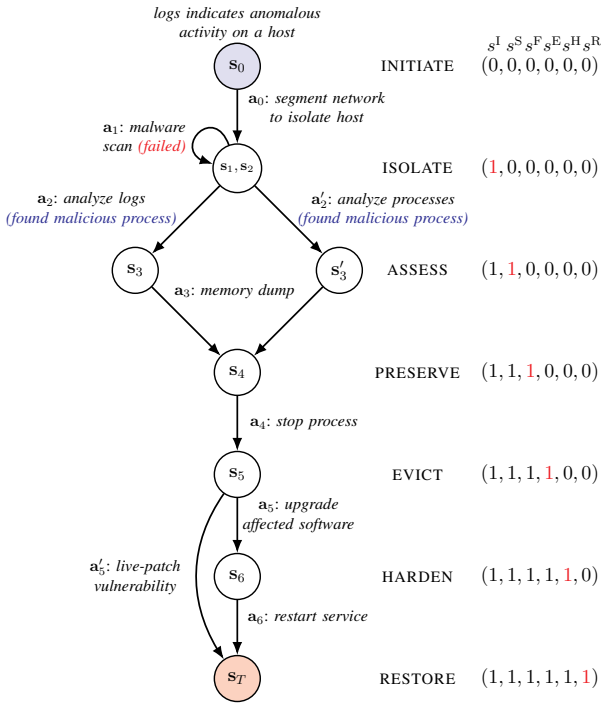


Fig. 5: Two example evolutions of the recovery state $\mathbf{s}_t$; cf. (3). The first recovery trajectory involves the actions $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6$ and the second trajectory involves the actions $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2', \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5'$.

This single action both isolates the attack ($s_t^{\mathrm{I}} = 1$) and revokes attacker access ($s_t^{\mathrm{E}} = 1$). In contrast, an advanced persistent threat (APT) typically requires multiple actions to complete these stages. For example, containment may involve isolating compromised hosts ($s_t^{\mathrm{I}} = 1$) and eviction may require malware removal or credential rotation ($s_t^{\mathrm{E}} = 1$). Thus, the recovery time T for an APT is typically longer than for a DOS attack.

Response generation. Because the recovery state contains information about the attacker, it is generally not known with certainty. However, the LLM can predict the state based on the available system logs and threat intelligence, which we denote by $\mathbf{I}$. Such predictions allow us to generate a response plan through auto-regressive sampling as follows. We start by generating the first action as $\mathbf{a}_0 \sim p_{\boldsymbol{\theta}'}(\cdot \mid \mathbf{s}_0, \mathbf{I})$, where the initial state is $\mathbf{s}_0 = (0, 0, 0, 0, 0, 0)$. Subsequently, we evaluate the effect of the action by predicting the next recovery state as $\tilde{\mathbf{s}}_1 \sim p_{\boldsymbol{\theta}'}(\cdot \mid \mathbf{s}_0, \mathbf{a}_0, \mathbf{I})$. We then repeat the same procedure to generate the next action as $\mathbf{a}_1 \sim p_{\boldsymbol{\theta}'}(\cdot \mid \tilde{\mathbf{s}}_1, \mathbf{I})$. This iterative procedure continues until the LLM predicts that the terminal recovery state $\tilde{\mathbf{s}}_t = (1, 1, 1, 1, 1, 1)$ has been reached.

Remark 4. *To instruct the* LLM *whether to generate an action or to predict the state, we append an instruction to the prompt. For brevity, we do not explicitly denote this instruction in the equations. Our prompt templates are available at [19].*

The expected time to recover from the incident when using response actions generated by the LLM depends on the current recovery state $\mathbf{s}_t$ (which captures the effects of previous actions) and the type of incident, as characterized by the vector $\mathbf{I}$. We formally define this recovery time-to-go as follows.

Definition 3 (Recovery time-to-go)**.** *Given an incident described by* $\mathbf{I}$*, the expected recovery time-to-go from the state* $\mathbf{s}$ *when executing actions generated by the* LLM $p_{\boldsymbol{\theta}'}$ *is*

$$J(\mathbf{s}) = \begin{cases} 0 & \text{if } \mathbf{s} = (1,1,1,1,1,1), \\ \mathbb{E}_{\mathbf{a}_t \sim p_{\boldsymbol{\theta}'}(\cdot|\tilde{\mathbf{s}}_t, \mathbf{I})} \{T \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{I}\} & \text{otherwise.} \end{cases}$$

Given this definition, we say that a response action is *hallucinated* if it has no effect on the expected recovery time-to-go. In other words, it does not contribute any progress toward recovery. This notion is formally defined below.

Definition 4 (Hallucinated response action)**.** *A response action* $\mathbf{a}_t$ *is hallucinated if it leads to a recovery state with the same expected recovery time-to-go as the current state, i.e.,*

$$J(\mathbf{s}_t) - \mathbb{E}_{\mathbf{s}_{t+1}} \{J(\mathbf{s}_{t+1}) \mid \mathbf{a}_t, \mathbf{s}_t, \mathbf{I}\} = 0, \quad \textit{for all } \mathbf{s}_t \in \tilde{\mathcal{S}},$$

where $\tilde{\mathcal{S}}$ *denotes the set of all states except* $(1,1,1,1,1,1)$.

This definition implies that hallucinations can be avoided by iteratively generating actions until one is found that reduces the expected recovery time-to-go. However, this approach to reducing hallucinations is not feasible in practice, as computing the recovery time requires knowledge of the attacker's behavior. An alternative approach is to involve a security operator to manually assess whether each action is hallucinated. While feasible, this approach defeats the purpose of using an LLM in the first place, namely to assist security operators.

To circumvent these limitations, we adopt a different approach, known as *self-verification* [65]. Following this approach, we *estimate* the recovery time-to-go of response actions using the LLM itself. This verification enforces a form of *self-consistency* [63], where actions are validated against the LLM's predicted outcomes. Such validations have been shown to reduce hallucinations (see e.g., [64] and [66]) and form the basis for our planning algorithm, as described below. (For a theoretical justification for why this approach can reduce the probability of hallucinated response actions, see §V.)

Planning algorithm. At each time t of the response, we use the LLM to generate N candidate actions $\mathcal{A}_t^N = \{\mathbf{a}_t^1, \ldots, \mathbf{a}_t^N\}$. Then, for each action $\mathbf{a}_t^i$, we use the LLM to simulate a *recovery trajectory* $\tilde{\mathbf{s}}_{t+1}, \mathbf{a}_{t+1} \ldots$, by sampling actions and updating the state until $\tilde{\mathbf{s}}_T = (1,1,1,1,1,1)$. We then use the length of the simulated trajectory as an estimate of the expected recovery time-to-go. We define this estimate as

$$\tilde{Q}(\tilde{\mathbf{s}}_t, \mathbf{a}_t^i) \approx 1 + \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} p_{\boldsymbol{\theta}'}(\mathbf{s}_{t+1} \mid \tilde{\mathbf{s}}_t, \mathbf{a}_t^i, \mathbf{I}) \tilde{J}(\mathbf{s}_{t+1}),$$

where $\tilde{J}$ is the estimated recovery time-to-go function.

Finally, we select the action with the shortest expected recovery time-to-go according to the estimate, i.e.,

$$\tilde{\mathbf{a}}_t \in \arg\min_{\mathbf{a}_t^i \in \mathcal{A}_t^N} \tilde{Q}(\tilde{\mathbf{s}}_t, \mathbf{a}_t^i). \tag{4}$$

This planning procedure is illustrated conceptually in Fig. 6 and the pseudocode is listed in Alg. 1. In the next section, we analyze the theoretical properties of this procedure and establish conditions under which it reduces hallucination. We also derive a bound on its hallucination probability.

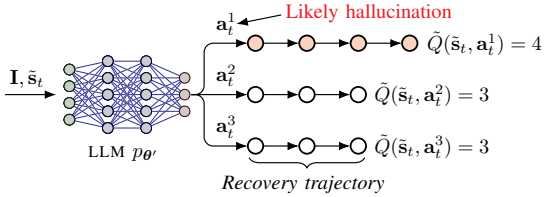


Fig. 6: Our planning procedure to circumvent hallucinations. At each time step of the response, we prompt the LLM with the incident description $\mathbf{I}$ and the current (predicted) recovery state $\tilde{\mathbf{s}}_t$ to generate N candidate actions (here $N = 3$). We then estimate the expected recovery time-to-go [denoted by $\tilde{Q}(\tilde{\mathbf{s}}_t, \mathbf{a}_t^i)$] of each action $\mathbf{a}_t^i$ by using the LLM to simulate possible recovery trajectories. If an action is hallucinated (per Def. 4), then it does not make progress toward recovery and thus leads to a longer recovery trajectory. Therefore, we select the action that leads to the shortest predicted recovery trajectory (either $\mathbf{a}_t^2$ or $\mathbf{a}_t^3$ in this example); cf. (4). This planning allows us to circumvent hallucinations under certain conditions, see §V-A for details.

Algorithm 1: Incident response planning with an LLM.

1 **Input:** LLM $p_{\boldsymbol{\theta}'}$, system logs $\mathbf{I}$, # actions N, # samples M.
2 **Output:** A response plan ρ, i.e., a sequence of response actions.
3 Initialize $\tilde{\mathbf{s}}_0 \leftarrow (0,0,0,0,0,0)$, $\rho \leftarrow \emptyset$, $t \leftarrow 0$.
4 **while** $\tilde{\mathbf{s}}_t \neq (1,1,1,1,1,1)$ **do**
5 Sample $\mathbf{a}_t^1, \ldots, \mathbf{a}_t^N$ from $p_{\boldsymbol{\theta}'}(\cdot \mid \tilde{\mathbf{s}}_t, \mathbf{I})$.
6 **for** $i = 1, 2, \ldots, N$ **do**
7 $\tilde{Q}(\tilde{\mathbf{s}}_t, \mathbf{a}_t^i) = \frac{1}{M} \sum_{k=1}^{M}$ RECOVERY-TIME$(\tilde{\mathbf{s}}_t, \mathbf{a}_t^i, \mathbf{I})$.
8 **end**
9 Select action $\tilde{\mathbf{a}}_t \in \arg\min_{\mathbf{a}_t^i} \tilde{Q}(\tilde{\mathbf{s}}_t, \mathbf{a}_t^i)$, $\rho \leftarrow \rho \cup \{(t, \tilde{\mathbf{a}}_t)\}$.
10 Update the state as $\tilde{\mathbf{s}}_{t+1} \sim p_{\boldsymbol{\theta}'}(\cdot \mid \tilde{\mathbf{s}}_t, \tilde{\mathbf{a}}_t, \mathbf{I})$, $t \leftarrow t + 1$.
11 **end**
12 **return** ρ.
13 **Procedure** RECOVERY-TIME $(\tilde{\mathbf{s}}, \mathbf{a}, \mathbf{I})$
14 Predict the state as $\tilde{\mathbf{s}}' \sim p_{\boldsymbol{\theta}'}(\cdot \mid \tilde{\mathbf{s}}, \mathbf{a}, \mathbf{I})$.
15 **if** $\tilde{\mathbf{s}}' = (1,1,1,1,1,1)$ **then**
16 **return** 1.
17 **end**
18 **else**
19 Sample $\mathbf{a}'$ from $p_{\boldsymbol{\theta}'}(\cdot \mid \tilde{\mathbf{s}}', \mathbf{I})$.
20 **return** $1 +$ RECOVERY-TIME$(\tilde{\mathbf{s}}', \mathbf{a}', \mathbf{I})$.
21 **end**
22 **end**

V. Analysis of the Hallucination Probability

To analyze the probability that our method generates a hallucinated response action, we distinguish between two cases: (*i*) at least one of the N candidate actions is non-hallucinated; and (*ii*) all N actions are hallucinated; cf. (4). In the following, we establish a sufficient condition under which hallucinations are avoided in case (*i*), and derive a probabilistic upper bound on the probability that case (*ii*) occurs.

A. Sufficient Conditions for Filtering Hallucinations

The purpose of the minimization (4) is to filter *hallucinated actions*, i.e., actions that do not affect the recovery time. This filtering is effective when the lookahead simulations [cf. lines 13–22 in Alg. 1] accurately reflect that hallucinated actions have no beneficial impact on the expected recovery time. However, because these simulations rely on the LLM to predict action outcomes, the filtering is inherently imperfect. Consequently, the effectiveness of the planning step [cf. (4)] depends on two key factors: (*i*) the degree to which hallucinated and non-hallucinated actions can be distinguished based on their impact on expected recovery time; and (*ii*) the accuracy of the LLM's predictions of the resulting recovery state $\mathbf{s}_t$; cf. Def. 1.

To quantify these two factors, let $\mathcal{A}$ denote the set of all possible response actions (as defined by the vocabulary of the LLM) and let $\mathcal{A}(\mathbf{s}, \mathbf{I})$ be the subset of non-hallucinated actions for the incident described by $\mathbf{I}$, given the recovery state $\mathbf{s}$. Moreover, let δ denote the minimal change in the recovery time-to-go when taking a non-hallucinated action, i.e.,

$$\delta = \min \left\{ J(\mathbf{s}_t) - \mathbb{E}_{\mathbf{s}_{t+1}} \left\{ J(\mathbf{s}_{t+1}) \mid \mathbf{a}, \mathbf{s}_t, \mathbf{I} \right\} \,\middle|\, \mathbf{a} \in \mathcal{A}(\mathbf{s}, \mathbf{I}) \right\}.$$

In view of Def. 4, we have $\delta > 0$.

Similarly, let η denote the total variation between the LLM's predictions and the true system dynamics (denoted by P), i.e.,

$$\sum_{\mathbf{s}' \in \mathcal{S}} \left| p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}, \mathbf{I}) - P(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}, \mathbf{I}) \right| \leq \eta, \quad \forall \mathbf{s} \in \tilde{\mathcal{S}}, \mathbf{a} \in \mathcal{A},$$

where $\mathcal{S}$ is the set of all recovery states and $\tilde{\mathcal{S}}$ is the set of non-terminal recovery states, i.e., $\tilde{\mathcal{S}} = \mathcal{S} \setminus \{(1,1,1,1,1,1)\}$. Note that the parameter η is upper bounded by 2, i.e., $0 \leq \eta \leq 2$.

Given the parameters δ and η, we have the following result.

Proposition 1. *Assuming that a) the number of sample trajectories M in Alg. 1 is sufficiently large so that the empirical mean approximates the true expectation and b) that both the expected recovery time and the* LLM*'s predicted recovery time are finite, i.e., $\|J\|_\infty < \infty$ and $\|\tilde{J}\|_\infty < \infty$. If at least one action in the set $\mathcal{A}_t^N$ [cf. (4)] is non-hallucinated and*

$$\delta > 2\eta \|J\|_\infty \left(\|\tilde{J}\|_\infty + 1 \right),$$

then the action selected by Alg. 1 will be non-hallucinated.

We present the proof of Prop. 1 in Appendix A. This proposition provides a sufficient condition under which the minimization (4) effectively filters hallucinated actions. The main condition of the proposition is that δ (which captures the degree to which hallucinations and non-hallucinations can be distinguished) is sufficiently large in comparison with the inaccuracy of the LLM's predictions, as quantified by η. While these parameters are likely unknown in practice, they can be estimated offline. For instance, δ can be estimated based on a curated set of incidents with known recovery outcomes. Similarly, η can be estimated by comparing the LLM's predictions with traces of historical incidents.

If at least one action in the set $\mathcal{A}_t^N$ [cf. (4)] would always be non-hallucinated, Prop. 1 would imply a condition that provides a guarantee of avoiding hallucinations. However, in practice, it is possible that all actions in $\mathcal{A}_t^N$ are hallucinated, in which case the lookahead minimization (4) will not help. We quantify the probability of this event in the next subsection.

B. Upper Bound on the Hallucination Probability

To complement the above condition for filtering hallucinations, we now analyze the *hallucination probability*. The main difficulty in this analysis is that the LLM's propensity to hallucinate is not known a priori. For this reason, we base our analysis on empirical observations of its behavior.

To obtain such empirical observations, we start by using the LLM to generate L sample actions. We then verify how many of those actions are hallucinated to estimate the LLM's hallucination probability h. We denote this estimate by $\overline{h}$. Due to sampling variability, this estimate may differ substantially from the probability h. To address this possibility, we establish a bound that quantifies how likely it is for the estimate $\overline{h}$ to deviate from the hallucination probability h by more than a given threshold ϵ, as stated in the following proposition.

Proposition 2. *Let h denote the true (but unknown) hallucination probability of the* LLM *and let $\overline{h}$ denote the empirical probability based on L samples. We have*

$$P(h \geq \overline{h} + \epsilon) \leq e^{-2\epsilon^2 L},$$

where $\epsilon > 0$ is a configurable parameter.

Proof. We model the process of generating L actions and verifying which of them are hallucinated as L independent and identically distributed Bernoulli trials, represented by the random variables $X_1, X_2, \ldots, X_L$. We have $X_i = 1$ if the ith sampled action is hallucinated; $X_i = 0$ otherwise. Hence, $\overline{h} = \frac{1}{L} \sum_{i=1}^{L} X_i$. Applying Hoeffding's inequality, we have

$$P(h \geq \overline{h} + \epsilon) \leq e^{-2\epsilon^2 L}.$$

□

This proposition implies that the probability that all actions in the set $\mathcal{A}_t^N$ [cf. (4)] are hallucinated (i.e., h^N) can be controlled with a certain confidence when the conditions of Prop. 1 hold by increasing N. Moreover, the confidence increases exponentially with the number of samples L used to estimate the hallucination probability, as shown in Fig. 7.

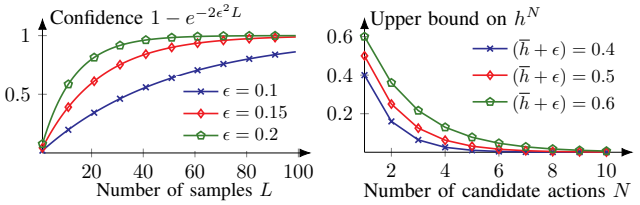


Fig. 7: Illustration of Prop. 2. Here L is the number of samples for estimating the hallucination probability and N is the number of candidate actions; cf. (4).

VI. Summary of Our Method

In summary, our method for using an LLM as decision support during incident handling consists of three main steps:

1) *Offline instruction fine-tuning of a lightweight* LLM.
   - We fine-tune the LLM via supervised learning on a dataset of logs from $68,000$ incidents paired with response plans and chain-of-thought reasoning steps.
2) *Online information retrieval.*
   - Before prompting the LLM with system logs to generate candidate response actions, we enrich the logs with threat intelligence retrieved from external sources.
3) *Online lookahead planning via Alg. 1.*
   - Instead of directly executing the action generated by the fine-tuned LLM, we generate several candidate actions and select the one that leads to the shortest predicted recovery time, which reduces the probability of hallucinations under certain conditions; cf. Prop. 1.

Dataset	*System*	*Attacks*	*Logs*
CTU-Malware-2014 [70]	WINDOWS XP SP2 servers	Various malwares and ransomwares, e.g., CRYPTODEFENSE [71].	SNORT alerts [72]
CIC-IDS-2017 [73]	WINDOWS and LINUX servers	Denial-of-service, web attacks, heartbleed, SQL injection, etc.	SNORT alerts [72]
AIT-IDS-V2-2022 [74]	LINUX and WINDOWS servers/hosts	Multi-stage attack with reconnaissance, cracking, and escalation.	WAZUH alerts [75]
CSLE-IDS-2024 [26]	LINUX servers	SAMBACRY, SHELLSHOCK, exploit of CVE-2015-1427, etc.	SNORT alerts [72]

TABLE 2: The datasets of cyberattacks and logs used for the experimental evaluation.

VII. Experimental Evaluation of Our Method

In this section, we present an experimental evaluation of our method. We start by comparing its performance with that of frontier LLMs based on log data from incidents reported in the literature. We then compare the performance of our method with that of a popular reinforcement learning method, namely proximal policy optimization (PPO) [18]. Our main evaluation metric is the recovery time T, as defined in Def. 2[1].

We instantiate our method with the DEEPSEEK-R1-14B LLM [61], which we fine-tune using the procedure described in §IV-B. Further, we implement the RAG pipeline described in §IV-C using the open threat exchange (OTX) API [76]. Finally, we instantiate the planning procedure described in Alg. 1 with $N = 3$ candidate actions and $M = 3$ samples; cf. (4). Additional experimental details and hyperparameters are provided in Appendices C–D. We provide source code, model parameters, and a video demonstration of our method in [19].

A. Comparison with Frontier LLMs

We compare our method with three frontier LLMs: DEEPSEEK-R1 [61], GEMINI 2.5 [16], and OPENAI O3 [14]. Compared to these models, the main difference is that our method is significantly more lightweight; see Table 3.

Method	*Number of parameters*	*Context window size*
OUR METHOD	14 billion	128,000
DEEPSEEK-R1 [61]	671 billion [61]	128,000
GEMINI 2.5 [16]	unknown ($\geq$ 100 billion)	1 million
OPENAI O3 [14]	unknown ($\geq$ 100 billion)	200,000

TABLE 3: Comparison between our method and frontier LLMs in terms of the number of model parameters and context window size.

Evaluation datasets. The evaluation is based on log data from 25 incidents across 4 different datasets published in the literature, namely CTU-Malware-2014 [70], CIC-IDS-2017 [73], AIT-IDS-V2-2022 [74], and CSLE-IDS-2024 [26]; see Table 2 and Fig. 8. We also include 5 false-positive incidents. Each incident contains log data and a brief system description. Given this data, the task of the LLM is to generate effective response actions, which we compare against the ground truth. The prompt templates and the datasets are available in [19].

We provide a (condensed) example of an incident and the ground truth response from the CTU-Malware-2014 dataset [70] below. In this example, the (ground truth) response plan consists of $T = 6$ response actions. Hence, the shortest possible recovery time an LLM can achieve when evaluated on this example is 6. However, if the LLM generates a plan that includes unnecessary response actions, then the recovery time will be longer than 6. It is also possible that the generated actions fail to fully recover the system from the incident. We report such cases separately in the evaluation results.

[1]To penalize unnecessary responses in the evaluation, we increment the recovery time by two instead of one if an action includes unnecessary steps.

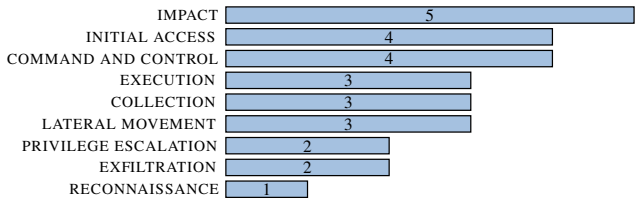


Fig. 8: Number of occurrences of different MITRE ATT&CK TACTICS [77] among the incidents in the evaluation datasets.

Example incident from the CTU-Malware-2014 dataset [70].

System description (condensed): 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 and configurations are...

Snort alert logs (condensed):

```
[120:3:2] (http_inspect) NO CONTENT-LENGTH..
[1:31033:6] MALWARE Win.Trojan.Cryptodefence..
{TCP} 147.32.84.165:1057 -> 222.88.205.195:443
[129:5:1] Bad segment, adjusted size..
[139:1:1] (spp_sdf) SDF..
```

Incident summary: Server 147.32.84.165 is infected with the WIN.TROJAN.CRYPTODEFENCE ransomware. Alerts show the server is making outbound command and control (C2) connections to 222.88.205.195. This indicates that the ransomware is active and may be preparing to encrypt files or has already begun doing so.

Response actions (condensed):

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. This preserves the ransomware executable, encrypted files, and other evidence for future analysis.

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 (which is obsolete) 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.

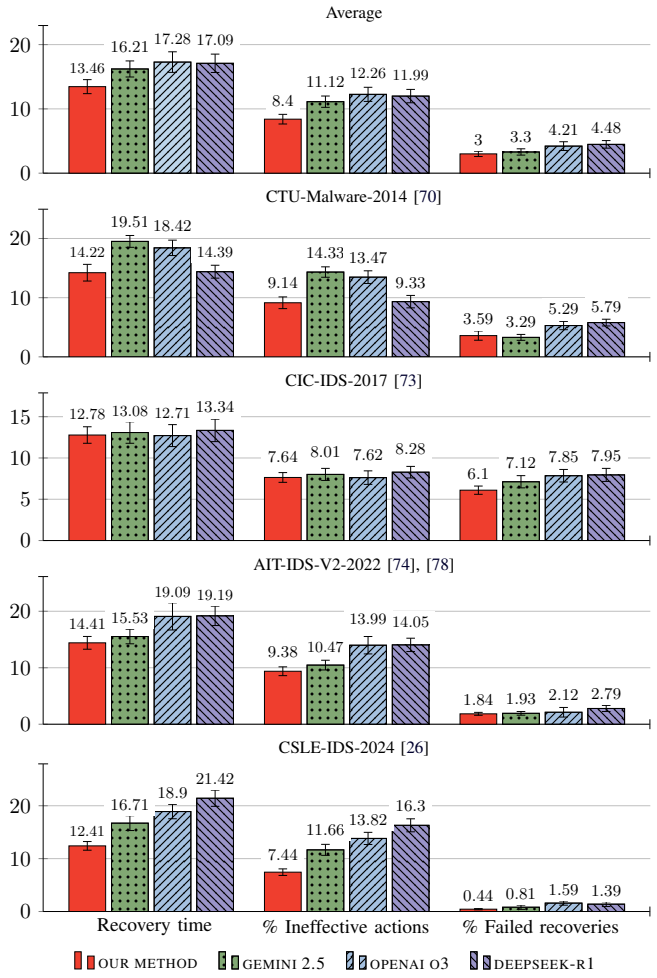


Fig. 9: Evaluation results (↓ better): comparison between our method and frontier LLMs. Bar colors relate to different methods; bar groups indicate performance metrics; numbers and error bars indicate the mean and the standard deviation from 5 evaluations with different random seeds.

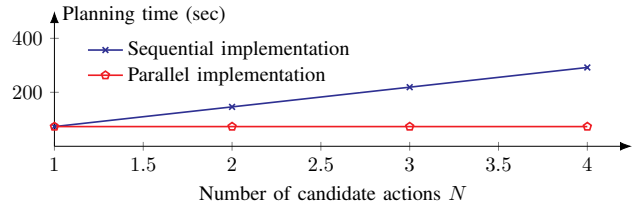


Fig. 10: Time required (per time step) to execute Alg. 1 for varying number of candidate actions N. The average planning times were computed based on 5 executions with RTX 8000 GPUs.

Evaluation results. The results are summarized in Table 4 and Fig. 9. Across all evaluation datasets, our method achieves the shortest average recovery time. On average, the recovery time of our method is 13.46 compared to 16.21 for the next best method. Among the frontier LLMs, we observe that GEMINI 2.5 performs best on average, whereas the difference between OPENAI O3 and DEEPSEEK-R1 is not statistically significant.

Scalability analysis. Figure 10 shows the compute time per time step of Alg. 1 for varying number of candidate actions N. We observe that the planning time increases linearly with N when the actions are evaluated sequentially. However, by parallelizing the computation across multiple GPUs, the planning time remains nearly constant as N increases.

Method	*Recovery time*	*% Ineffective actions*	*% Failed recovery*
Average			
OUR METHOD	$\mathbf{13.46 \pm 1.09}$	$\mathbf{8.40 \pm 0.75}$	$\mathbf{3.00 \pm 0.41}$
GEMINI 2.5 [16]	16.21 ± 1.25	11.12 ± 0.88	3.30 ± 0.49
OPENAI O3 [14]	17.28 ± 1.60	12.26 ± 1.10	4.21 ± 0.68
DEEPSEEK-R1 [61]	17.09 ± 1.43	11.99 ± 1.04	4.48 ± 0.59
CTU-Malware-2014 [70]			
OUR METHOD	$\mathbf{14.22 \pm 1.41}$	$\mathbf{9.14 \pm 0.99}$	3.59 ± 0.78
GEMINI 2.5 [16]	19.51 ± 1.00	14.33 ± 0.88	$\mathbf{3.29 \pm 0.50}$
OPENAI O3 [14]	18.42 ± 1.30	13.47 ± 1.07	5.29 ± 0.70
DEEPSEEK-R1 [61]	14.39 ± 1.09	9.33 ± 1.05	5.79 ± 0.57
CIC-IDS-2017 [73]			
OUR METHOD	12.78 ± 1.00	7.64 ± 0.59	$\mathbf{6.11 \pm 0.50}$
GEMINI 2.5 [16]	13.08 ± 1.30	8.01 ± 0.73	7.13 ± 0.74
OPENAI O3 [14]	$\mathbf{12.71 \pm 1.33}$	$\mathbf{7.62 \pm 0.82}$	7.86 ± 0.76
DEEPSEEK-R1 [61]	13.34 ± 1.36	8.28 ± 0.70	7.95 ± 0.79
AIT-IDS-V2-2022 [74], [78]			
OUR METHOD	$\mathbf{14.41 \pm 1.13}$	$\mathbf{9.38 \pm 0.78}$	$\mathbf{1.84 \pm 0.26}$
GEMINI 2.5 [16]	15.53 ± 1.29	10.47 ± 0.86	1.94 ± 0.33
OPENAI O3 [14]	19.09 ± 2.40	13.99 ± 1.54	2.12 ± 0.85
DEEPSEEK-R1 [61]	19.19 ± 1.72	14.05 ± 1.18	2.79 ± 0.54
CSLE-IDS-2024 [26]			
OUR METHOD	$\mathbf{12.41 \pm 0.82}$	$\mathbf{7.44 \pm 0.62}$	$\mathbf{0.44 \pm 0.11}$
GEMINI 2.5 [16]	16.71 ± 1.40	11.66 ± 1.04	0.81 ± 0.40
OPENAI O3 [14]	18.90 ± 1.36	13.82 ± 1.15	1.59 ± 0.39
DEEPSEEK-R1 [61]	21.42 ± 1.53	16.30 ± 1.23	1.39 ± 0.45

TABLE 4: Evaluation results: comparison between our method and frontier LLMs. Rows relate to different methods; columns indicate performance metrics (↓ better); blue rows relate to our method (see Fig. 1); the best results are highlighted in bold; numbers indicate the mean and the standard deviation from 5 evaluations with different random seeds.

Hallucination analysis. Figure 11 shows the empirical hallucination probability of our method based on $L = 30$ response actions sampled from the LLM when prompted with log data from the evaluation datasets. The figure also shows the theoretical upper bound expressed in Prop. 2.

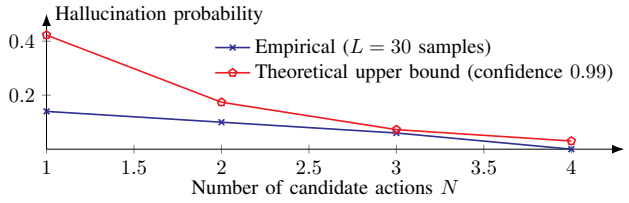


Fig. 11: The empirical hallucination probability of our method for varying number of candidate actions N, as well as the theoretical upper bound on the hallucination probability h^N (assuming the conditions of Prop. 1 hold) with confidence 0.99, i.e., the right-hand side of the bound in Prop. 2 is 0.01.

We observe in Fig. 11 that the theoretical bound holds uniformly over the empirical probabilities. However, while the theoretical bound decreases exponentially with the number of candidate actions N, the empirical hallucination probability exhibits an approximately linear decline. This discrepancy suggests that the conditions of Prop. 1 are not fully satisfied. This is expected, as we used only $M = 3$ samples to estimate the expected recovery times. We chose this value of M to ensure that the planning could be completed within

a reasonable time using our limited hardware (QUADRO RTX 8000 GPUS). Increasing M is likely to yield more accurate estimates and further reduce the probability of hallucinations.

Ablation study. To evaluate the relative importance of each step of our method (i.e., fine-tuning, RAG, and planning), we evaluate our method with and without each step. The results are summarized in Table 5 and Fig. 12. We observe consistent performance degradations when each step is removed. The most substantial degradation occurs when fine-tuning is removed, which causes the average recovery time to increase from 13 to 25. Planning also has a significant impact. Without planning, the average recovery time jumps from 13 to 21. Retrieval augmented generation (RAG) contributes as well, though its effects on the performance are more modest.

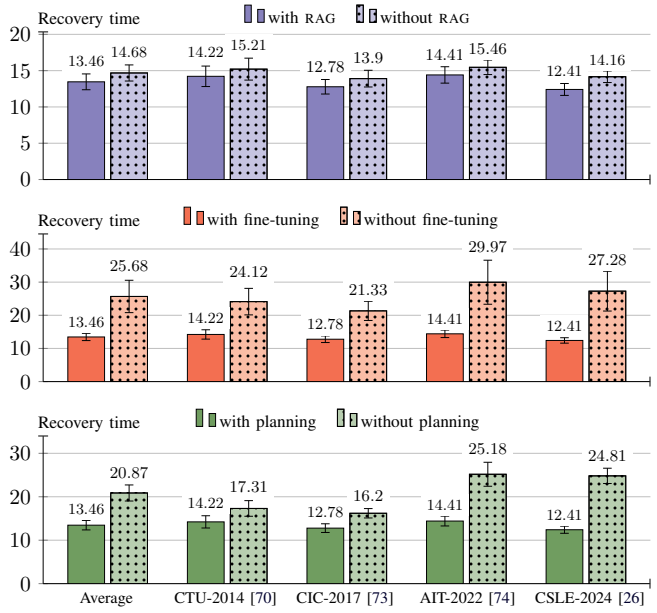


Fig. 12: Ablation-study results for the recovery time metric (↓ better). Bars relate to our method with and without different steps; bar groups indicate the evaluation dataset; numbers and error bars indicate the mean and the standard deviation from 5 evaluations with different random seeds.

Method	*Recovery time*	*% Ineffective actions*	*% Failed recovery*
	Average		
OUR METHOD	$\mathbf{13.46 \pm 1.09}$	$\mathbf{8.40 \pm 0.75}$	$\mathbf{3.00 \pm 0.41}$
without RAG	14.68 ± 1.11	9.58 ± 0.96	4.13 ± 0.44
without fine-tuning	25.68 ± 4.88	20.48 ± 3.51	12.64 ± 2.42
without planning	20.87 ± 1.85	15.91 ± 1.79	7.75 ± 0.79
	CTU-Malware-2014 [70]		
OUR METHOD	$\mathbf{14.22 \pm 1.41}$	$\mathbf{9.14 \pm 0.99}$	$\mathbf{3.59 \pm 0.78}$
without RAG	15.21 ± 1.51	10.20 ± 1.27	5.90 ± 0.84
without fine-tuning	24.12 ± 4.00	19.19 ± 2.87	12.12 ± 2.50
without planning	17.31 ± 1.80	12.35 ± 1.61	10.93 ± 1.03
	CIC-IDS-2017 [73]		
OUR METHOD	$\mathbf{12.78 \pm 1.00}$	$\mathbf{7.64 \pm 0.59}$	$\mathbf{6.11 \pm 0.50}$
without RAG	13.90 ± 1.15	8.67 ± 0.88	7.71 ± 0.61
without fine-tuning	21.33 ± 2.89	16.08 ± 2.26	13.87 ± 1.88
without planning	16.20 ± 1.08	11.64 ± 1.15	9.86 ± 0.56
	AIT-IDS-V2-2022 [74], [78]		
OUR METHOD	$\mathbf{14.41 \pm 1.13}$	$\mathbf{9.38 \pm 0.78}$	$\mathbf{1.84 \pm 0.26}$
without RAG	15.46 ± 1.00	10.33 ± 0.87	2.16 ± 0.20
without fine-tuning	29.97 ± 6.64	24.52 ± 4.35	14.98 ± 2.83
without planning	25.18 ± 2.76	20.01 ± 2.47	5.39 ± 1.02
	CSLE-IDS-2024 [26]		
OUR METHOD	$\mathbf{12.41 \pm 0.82}$	$\mathbf{7.44 \pm 0.62}$	$\mathbf{0.44 \pm 0.11}$
without RAG	14.16 ± 0.80	9.09 ± 0.84	0.76 ± 0.10
without fine-tuning	27.28 ± 6.00	22.12 ± 4.56	10.71 ± 2.50
without planning	24.81 ± 1.76	19.64 ± 1.92	4.82 ± 0.54

TABLE 5: Ablation-study results. Rows relate to different methods; columns indicate performance metrics (↓ better); blue rows relate to our method (see Fig. 1); the best results are highlighted in bold; numbers indicate the mean and the standard deviation from 5 evaluations with different random seeds.

B. Comparison with Proximal Policy Optimization

Numerous reinforcement learning approaches have been proposed for incident response, including policy optimization methods [33], tree search [34], stochastic approximation [79], and Q-learning [36]; see [80] for an extensive review of the state of the art. Among these methods, variants of proximal policy optimization (PPO) [18] dominate recent work. We therefore use PPO as a representative baseline for comparison.

Experiment setup. We evaluate our method against PPO on two simulated incidents: an advanced persistent threat from the CAGE-2 simulation [31], and a network intrusion scenario from [79]. The evaluation uses the same metrics as in the comparison with frontier LLMs[2]. The hyperparameters of PPO and the prompt templates that we use for the evaluation are available in Appendix C and [19], respectively.

Evaluation results. The results are presented in Fig. 13. Both our method and PPO achieve similar performance in terms of recovery time and failed recoveries across the two simulations. The only notable performance gap is in the percentage of ineffective actions for the CAGE-2 simulation, where our method performs better. The key difference between our method and PPO lies in their training requirements: PPO requires incident-specific training (approximately 10–20 minutes of training per incident) to reach good performance. In contrast, our method does not require such training to achieve good performance.

C. Discussion of the Evaluation Results

Our experimental results demonstrate a trade-off between generality, computational cost, and deployment practicality. Compared to frontier LLMs, our method is significantly more lightweight, i.e., it requires fewer parameters and runs efficiently on commodity hardware, yet it achieves consistently better performance across all evaluation metrics. This performance advantage is primarily driven by our fine-tuning and planning steps, as shown in the ablation study.

When compared to reinforcement learning methods such as PPO, our method is more computationally costly at inference time due to the overhead of planning; see Fig. 10. However, this overhead is offset by a major advantage: our method requires no incident-specific training. In contrast, PPO must be retrained for each new incident, which is impractical.

[2]To align the CAGE-2 scenario with our evaluation metrics, we exclude decoy-related actions as they target prevention rather than response.

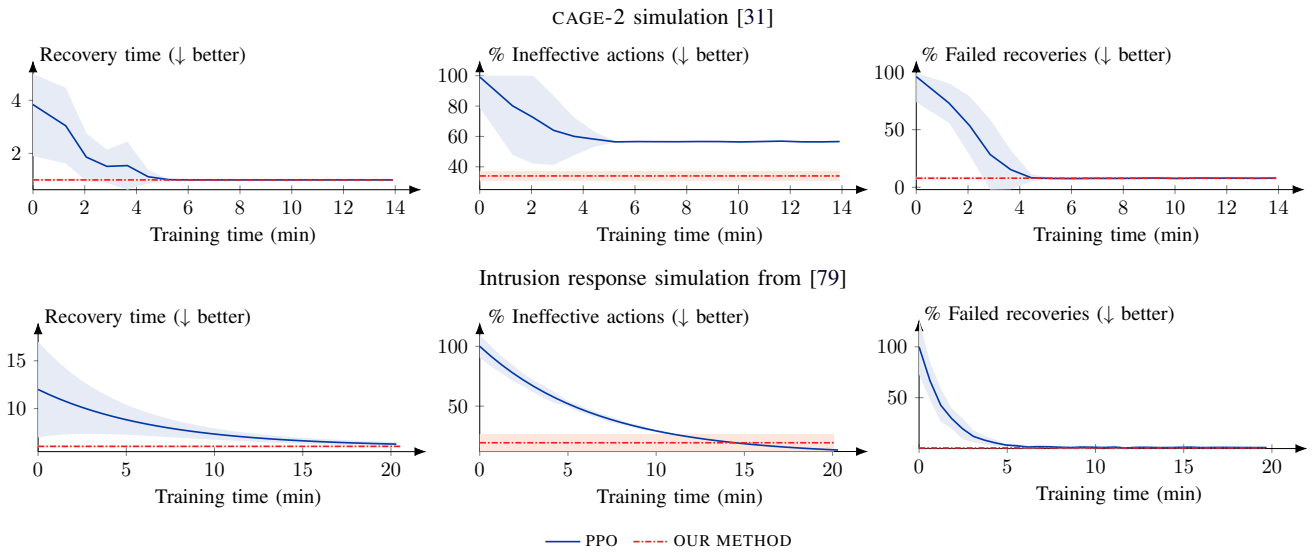


Fig. 13: Comparison between our method (red curves) and the PPO reinforcement learning method (blue curves) [18]. The first row of plots relates to the CAGE-2 simulation [31] and the second row relates to the network intrusion simulation from [79]. Columns relate to different evaluation metrics (↓ better). Curves show the mean value from evaluations with 5 random seeds; shaded areas indicate standard deviations. The x-axes indicate the training time required by PPO for each simulation. In contrast, our method requires no incident-specific training.

Compared to incident response playbooks [22], our method provides two clear advantages. First, it does not rely on domain experts for configuration. Second, it generates more precise and context-specific response actions. In particular, current playbooks often prescribe vague actions that are not directly executable [3], [4]. By contrast, our method produces executable response actions tailored to the system logs.

On the other hand, the main concern of our method compared to playbooks is the risk of hallucination. While our method reduces this risk through fine-tuning, information retrieval, and planning, it does not eliminate it entirely. Hence, response actions generated by our method should be subject to human validation before execution in most cases.

Takeaways. In summary, our main experimental findings are:

- Our method consistently outperforms frontier LLMs across all evaluation metrics, while being significantly more lightweight and able to run on commodity hardware.
- Fine-tuning and decision-theoretic planning are key drivers of performance, RAG is less important.
- Compared to reinforcement learning methods, our method has higher overhead but avoids incident-specific training.
- In contrast to response playbooks, our method does not rely on domain experts for configuration and generates more precise and actionable response plans.

VIII. Conclusion

We introduce a novel method that enables the effective use of a large language model (LLM) to provide decision support for incident response planning. Our method uses the LLM for translating system logs into effective response plans while addressing its limitations through fine-tuning, information retrieval, and decision-theoretic planning. We prove that our method produces incident responses with bounded hallucination probability; see Prop. 1 and Prop. 2. Under certain assumptions, this bound can be made arbitrarily small at the expense of increased planning time. We evaluate our method on logs from incidents reported in the literature. The results show that our method a) achieves up to 22% shorter recovery times than frontier LLMs and b) generalizes to a broad range of incident types and response actions.

Future work. A primary direction for future work is to conduct evaluations in operational settings, where security operators use our method for decision support. Such studies would provide insights into the practical utility of our method and how to improve it further. From a theoretical standpoint, a promising direction of future work is to tighten the hallucination-probability bound stated in Prop. 2. A possible approach to tighten this bound is to leverage conformal-abstention techniques [81]. Another direction for future work is to extend the system model in §IV-D to include additional performance metrics beyond recovery time. Moreover, due to the generality of our method, it is possible to extend our planning procedure [cf. Alg. 1] in many ways, e.g., by incorporating rollout techniques [82] or tree search [34]. Similarly, the information-retrieval step of our method can be expanded to integrate information from several sources.

Acknowledgment

This research is supported by the Swedish Research Council under contract 2024-06436.

Appendix A
Proof of Proposition 1

We start by noting that the planning problem in §IV-D can be viewed as a stochastic shortest path problem on the graph of recovery states, where the goal is to reach the state $\mathbf{s}_t = (1, 1, 1, 1, 1, 1)$ as quickly as possible. Consequently, the problem is well-defined under standard assumptions, see e.g.,

[83] for details. The main approach for proving Prop. 1 is to bound the difference in estimated recovery time of a non-hallucinated action and a hallucinated action. To this end, we start by stating and proving the following lemma.

Lemma 1. *Given the conditions of Prop. 1, we have*

$$\|\tilde{J} - J\|_\infty \leq \eta \|\tilde{J}\|_\infty \|J\|_\infty,$$

where J is the recovery time-to-go function [cf. Def. 3] and $\tilde{J}$ is the time-to-go function estimated by the LLM.

Proof. We first note that the vocabulary (i.e., the set of tokens) of any LLM is finite. Therefore, the set of feasible response actions $\mathcal{A}$ is finite. As a consequence, the state predictions $p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}, \mathbf{I})$ define a transition probability matrix between (non-terminal) recovery states. We denote this matrix by $\tilde{\mathbf{F}}$ and the corresponding matrix of the real system by $\mathbf{F}$, where $\tilde{\mathbf{F}}_{\mathbf{ss}'}$ denotes the transition probability between the non-terminal states $\mathbf{s}$ and $\mathbf{s}'$. Similarly, we use $\mathbf{J}$ and $\tilde{\mathbf{J}}$ to denote the vectors obtained by applying the functions J and $\tilde{J}$ to the set of non-terminal recovery states $\tilde{\mathcal{S}}$, i.e., all states for which $\mathbf{s} \neq (1,1,1,1,1,1)$, where $\mathbf{J}_\mathbf{s}$ denotes the expected recovery time-to-go from the non-terminal state $\mathbf{s}$.

Since the goal is to minimize the recovery time, we can express the recovery time-to-go function recursively by defining a stage cost of 1 for each response action taken. Using this formulation of the recovery time-to-go, we have

$$\begin{aligned} \mathbf{J} &= \mathbf{1} + \mathbf{F}\mathbf{J}, \\ \tilde{\mathbf{J}} &= \mathbf{1} + \tilde{\mathbf{F}}\tilde{\mathbf{J}}, \end{aligned} \tag{5}$$

where $\mathbf{1}$ denotes the vector of all ones. Given these Bellman equations, the difference $\tilde{\mathbf{J}} - \mathbf{J}$ can be written as

$$\begin{aligned} \tilde{\mathbf{J}} - \mathbf{J} &= \left(\mathbf{1} + \tilde{\mathbf{F}}\tilde{\mathbf{J}}\right) - (\mathbf{1} + \mathbf{F}\mathbf{J}) \\ &= \tilde{\mathbf{F}}\tilde{\mathbf{J}} - \mathbf{F}\mathbf{J} \\ &= \tilde{\mathbf{F}}\tilde{\mathbf{J}} - \mathbf{F}\mathbf{J} + \tilde{\mathbf{F}}\mathbf{J} - \tilde{\mathbf{F}}\mathbf{J} \\ &= \tilde{\mathbf{F}}(\tilde{\mathbf{J}} - \mathbf{J}) + (\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J}. \end{aligned}$$

Solving for $(\tilde{\mathbf{J}} - \mathbf{J})$ gives

$$\begin{aligned} &\tilde{\mathbf{J}} - \mathbf{J} = \tilde{\mathbf{F}}(\tilde{\mathbf{J}} - \mathbf{J}) + (\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J} \\ \implies &(\mathbb{1} - \tilde{\mathbf{F}})(\tilde{\mathbf{J}} - \mathbf{J}) = (\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J} \\ \implies &\tilde{\mathbf{J}} - \mathbf{J} = (\mathbb{1} - \tilde{\mathbf{F}})^{-1}(\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J}, \end{aligned} \tag{6}$$

where $\mathbb{1}$ denotes the identity matrix.

Since $\|J\|_\infty$ and $\|\tilde{J}\|_\infty$ are assumed finite, $\|\mathbf{J}\|_\infty$ and $\|\tilde{\mathbf{J}}\|_\infty$ are also finite. As a consequence, the linear systems in (5) have unique solutions. Consequently, the inverse in (6) exists. Taking the supremum norm on both sides of the final expression in (6), we have

$$\begin{aligned} \|\tilde{\mathbf{J}} - \mathbf{J}\|_\infty &= \|(\mathbb{1} - \tilde{\mathbf{F}})^{-1}(\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J}\|_\infty \\ &\leq \|(\mathbb{1} - \tilde{\mathbf{F}})^{-1}\|_\infty \|(\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J}\|_\infty, \end{aligned} \tag{7}$$

where the last inequality follows from the sub-multiplicative property of the supremum norm. Hence, it suffices to show that the right-hand side in (7) is bounded by $\eta\|\tilde{J}\|_\infty\|J\|_\infty$. We start by showing that $\|(\mathbb{1} - \tilde{\mathbf{F}})^{-1}\|_\infty = \|\tilde{\mathbf{J}}\|_\infty$. In view of (5), we have

$$\begin{aligned} &\tilde{\mathbf{J}} = \mathbf{1} + \tilde{\mathbf{F}}\tilde{\mathbf{J}} \\ \implies &(\mathbb{1} - \tilde{\mathbf{F}})\tilde{\mathbf{J}} = \mathbf{1} \\ \implies &\tilde{\mathbf{J}} = (\mathbb{1} - \tilde{\mathbf{F}})^{-1}\mathbf{1} \\ \implies &\|\tilde{\mathbf{J}}\|_\infty = \|(\mathbb{1} - \tilde{\mathbf{F}})^{-1}\mathbf{1}\|_\infty. \end{aligned}$$

Since $\|\tilde{J}\|_\infty$ is assumed finite, the expected time to reach the terminal state $\mathbf{s} = (1,1,1,1,1,1)$ from any non-terminal state $\mathbf{s} \in \tilde{\mathcal{S}}$ is finite. As a consequence, the spectral radius of the transition matrix between the non-terminal states, i.e., $\tilde{\mathbf{F}}$, must be strictly less than 1. Therefore, we can expand $(\mathbb{1} - \tilde{\mathbf{F}})^{-1}$ using the Neumann series representation as

$$(\mathbb{1} - \tilde{\mathbf{F}})^{-1} = \sum_{k=0}^{\infty} \tilde{\mathbf{F}}^k.$$

Because the matrix $\tilde{\mathbf{F}}$ is non-negative, all of its powers are also non-negative. As a consequence, $(\mathbb{1} - \tilde{\mathbf{F}})^{-1}$ is non-negative. For any non-negative matrix $\mathbf{A}$, we have $\|\mathbf{A}\|_\infty = \|\mathbf{A}\mathbf{1}\|_\infty$. Consequently, we obtain

$$\begin{aligned} \|\tilde{\mathbf{J}}\|_\infty &= \|(\mathbb{1} - \tilde{\mathbf{F}})^{-1}\mathbf{1}\|_\infty \\ &= \|(\mathbb{1} - \tilde{\mathbf{F}})^{-1}\|_\infty. \end{aligned} \tag{8}$$

Now consider the second factor in the right-hand side of (7), i.e., $\|(\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J}\|_\infty$. Fix any recovery state $\mathbf{s}$. We have

$$\begin{aligned} \left|\left(\left(\tilde{\mathbf{F}} - \mathbf{F}\right)\mathbf{J}\right)_\mathbf{s}\right| &= \left|\sum_{\mathbf{s}' \in \tilde{\mathcal{S}}} \left(\tilde{\mathbf{F}}_{\mathbf{ss}'} - \mathbf{F}_{\mathbf{ss}'}\right)\mathbf{J}_{\mathbf{s}'}\right| \\ &\overset{(a)}{\leq} \sum_{\mathbf{s}' \in \tilde{\mathcal{S}}} \left|\tilde{\mathbf{F}}_{\mathbf{ss}'} - \mathbf{F}_{\mathbf{ss}'}\right| \cdot |\mathbf{J}_{\mathbf{s}'}| \\ &\leq \left(\sum_{\mathbf{s}' \in \tilde{\mathcal{S}}} \left|\tilde{\mathbf{F}}_{\mathbf{ss}'} - \mathbf{F}_{\mathbf{ss}'}\right|\right) \|\mathbf{J}\|_\infty \\ &\leq \eta \|\mathbf{J}\|_\infty, \end{aligned}$$

where we use the triangle inequality to move the absolute value inside the sum and then the fact that $|ab| = |a||b|$ to obtain (a). Since this bound holds for any state $\mathbf{s}$, we have

$$\|(\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J}\|_\infty \leq \eta \|\mathbf{J}\|_\infty.$$

Substituting this bound and (8) into (7) yields

$$\begin{aligned} \|\tilde{\mathbf{J}} - \mathbf{J}\|_\infty &\leq \|(\mathbb{1} - \tilde{\mathbf{F}})^{-1}\|_\infty \|(\tilde{\mathbf{F}} - \mathbf{F})\mathbf{J}\|_\infty \\ &\leq \eta \|\tilde{\mathbf{J}}\|_\infty \|\mathbf{J}\|_\infty. \end{aligned}$$

Since the recovery time-to-go from the terminal state is 0 [cf. Def. 3], we have $\|\tilde{\mathbf{J}} - \mathbf{J}\|_\infty = \|\tilde{J} - J\|_\infty$, $\|\tilde{\mathbf{J}}\|_\infty = \|\tilde{J}\|_\infty$, and $\|\mathbf{J}\|_\infty = \|J\|_\infty$. The proof is thus complete. □

Given Lemma 1, we are now ready to derive the proof of Prop. 1. The event that a hallucinated action $\hat{\mathbf{a}}$ is selected over a non-hallucinated action $\tilde{\mathbf{a}}$ in (4) implies

$$\tilde{Q}(\tilde{\mathbf{s}}, \hat{\mathbf{a}}) \leq \tilde{Q}(\tilde{\mathbf{s}}, \tilde{\mathbf{a}}).$$

To show that this inequality cannot hold under the propo-

sition's assumptions, we start by bounding the difference between $\tilde{Q}$ and Q, where $Q(\mathbf{s},\mathbf{a})$ is the true expected recovery time-to-go when taking response action $\mathbf{a}$ in state $\mathbf{s}$ and $\tilde{Q}(\mathbf{s},\mathbf{a})$ is the LLM's estimate. We have

$$
\begin{aligned}
&|\tilde{Q}(\mathbf{s},\mathbf{a}) - Q(\mathbf{s},\mathbf{a})| = \\
&\left| \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})\tilde{J}(\mathbf{s}') - \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} P(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})J(\mathbf{s}') \right| \\
&= \Bigg| \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})\tilde{J}(\mathbf{s}') - \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} P(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})J(\mathbf{s}') + \\
&\qquad \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})J(\mathbf{s}') - \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})J(\mathbf{s}') \Bigg| \\
&= \Bigg| \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) \Big(\tilde{J}(\mathbf{s}') - J(\mathbf{s}')\Big) - \\
&\qquad \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} \Big(P(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) - p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})\Big)J(\mathbf{s}') \Bigg| \\
&\overset{(a)}{\leq} \left| \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) \Big(\tilde{J}(\mathbf{s}') - J(\mathbf{s}')\Big) \right| + \\
&\qquad \left| \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} \Big(P(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) - p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})\Big)J(\mathbf{s}') \right| \\
&\overset{(b)}{\leq} \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} \left| p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) \Big(\tilde{J}(\mathbf{s}') - J(\mathbf{s}')\Big) \right| + \\
&\qquad \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} \left| \Big(P(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) - p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I})\Big)J(\mathbf{s}') \right| \\
&\overset{(c)}{\leq} \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} \left| p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) \Big(\tilde{J}(\mathbf{s}') - J(\mathbf{s}')\Big) \right| + \\
&\qquad \sum_{\mathbf{s}'\in\tilde{\mathcal{S}}} \left| p_{\boldsymbol{\theta}'}(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) - P(\mathbf{s}' \mid \mathbf{s},\mathbf{a},\mathbf{I}) \right| \|J\|_\infty \\
&\leq \|\tilde{J} - J\|_\infty + \eta\|J\|_\infty \\
&\overset{(d)}{\leq} \underbrace{\eta\|\tilde{J}\|_\infty\|J\|_\infty + \eta\|J\|_\infty}_{=\Delta},
\end{aligned}
$$

where (a)-(b) follow from the triangle inequality; (c) uses the fact that $\|\mathbf{ab}\|_\infty \leq \|\mathbf{a}\|_\infty\|\mathbf{b}\|_\infty$; and (d) follows from Lemma 1. This bound implies that

$$Q(\mathbf{s},\mathbf{a}) - \Delta \leq \tilde{Q}(\mathbf{s},\mathbf{a}) \leq Q(\mathbf{s},\mathbf{a}) + \Delta. \tag{9}$$

Now, if a hallucinated action $\hat{\mathbf{a}}$ is selected over a non-hallucinated action $\tilde{\mathbf{a}}$ in (4), we must have

$$\tilde{Q}(\mathbf{s},\hat{\mathbf{a}}) \leq \tilde{Q}(\mathbf{s},\tilde{\mathbf{a}}).$$

Combining this inequality with (9), we have

$$
\begin{aligned}
Q(\mathbf{s},\hat{\mathbf{a}}) - \Delta &\leq \tilde{Q}(\mathbf{s},\hat{\mathbf{a}}) \\
&\leq \tilde{Q}(\mathbf{s},\tilde{\mathbf{a}}) \\
&\leq Q(\mathbf{s},\tilde{\mathbf{a}}) + \Delta \\
\implies Q(\mathbf{s},\hat{\mathbf{a}}) - Q(\mathbf{s},\tilde{\mathbf{a}}) &\leq 2\Delta.
\end{aligned} \tag{10}
$$

Notation(s)	*Description*
$\mathbf{a}$	Response action; cf. §IV-D.
$\mathbf{s}, \tilde{\mathbf{s}}$	Recovery state [cf. (3)] and predicted state.
$\mathbf{I}$	Initial information about the incident (e.g., logs).
T	Recovery time; cf. §IV-D.
$p_{\boldsymbol{\theta}}, \boldsymbol{\theta}$	the token distribution of an LLM and its parameters; cf. (1).
$\boldsymbol{\theta}'$	Fine-tuned parameter vector of an LLM; cf. §IV-B.
$\mathcal{D}$	Instruction dataset for fine-tuning; cf. §IV-B.
$\mathbf{x}, \mathbf{y}$	Instruction and desired output; cf. §IV-B.
N	Number of candidate actions to evaluate; cf. §IV-D.
M	Number of samples to estimate expected values in Alg. 1.
$\tilde{\mathbf{a}}_t$	The response action selected after planning; cf. (4).
$J, \tilde{J}$	Recovery time-to-go functions (true and estimated); cf.§IV-D.
$Q, \tilde{Q}$	Q-functions (true and estimated by LLM); cf. §IV-D.
$\mathcal{S}, \mathcal{A}$	Sets of recovery states and response actions; cf. §IV-D.
$\tilde{\mathcal{S}}$	Set of non-terminal recovery states; cf. §IV-D.
$\mathcal{A}_t^N$	The set of N candidate actions at time step t; cf. §IV-D.

TABLE 6: Notation.

Next, since $\hat{\mathbf{a}}$ is hallucinated and $\tilde{\mathbf{a}}$ is not, we have

$$
\begin{aligned}
Q(\mathbf{s},\hat{\mathbf{a}}) &= 1 + \mathbb{E}_{\mathbf{s}'}[J(\mathbf{s}') \mid \hat{\mathbf{a}},\mathbf{s},\mathbf{I}] = 1 + J(\mathbf{s}), \\
Q(\mathbf{s},\tilde{\mathbf{a}}) &= 1 + \mathbb{E}_{\mathbf{s}'}[J(\mathbf{s}') \mid \tilde{\mathbf{a}},\mathbf{s},\mathbf{I}] \leq 1 + J(\mathbf{s}) - \delta.
\end{aligned}
$$

Substituting $Q(\mathbf{s},\hat{\mathbf{a}}) = 1 + J(\mathbf{s})$ into the inequality, we obtain

$$
\begin{aligned}
Q(\mathbf{s},\hat{\mathbf{a}}) &\geq Q(\mathbf{s},\tilde{\mathbf{a}}) + \delta \\
\implies \delta &\leq Q(\mathbf{s},\hat{\mathbf{a}}) - Q(\mathbf{s},\tilde{\mathbf{a}}) \leq 2\Delta \\
&= 2\left(\eta\|\tilde{J}\|_\infty\|J\|_\infty + \eta\|J\|_\infty\right) \\
&= 2\eta\|J\|_\infty\left(\|\tilde{J}\|_\infty + 1\right).
\end{aligned}
$$

Since the conditions of the proposition state that

$$\delta > 2\eta\|J\|_\infty\left(\|\tilde{J}\|_\infty + 1\right),$$

we conclude that whenever a non-hallucinated action exists, it will be selected by the minimization (4). □

APPENDIX B
NOTATION

Our notation is summarized in Table 6.

APPENDIX C
EXPERIMENTAL SETUP

All computations are performed using 4×QUADRO RTX 8000 GPUS. The hyperparameters that we use for fine-tuning and for instantiating PPO are listed in Table 7. Parameters not listed in Table 7 are set to default values.

APPENDIX D
DATASET GENERATION

To generate the dataset of instruction-answer pairs for fine-tuning, we use a combination of log data from our testbed and synthetic data generated by frontier LLMs. Specifically, we first run the attacks listed in Table 8 on our testbed. Such runs generate system measurements and logs (e.g., SNORT alerts [72]), based on which we construct 500 instruction-answer pairs. Each instruction consists of log data and previously applied response actions, as well as a task to either generate a response action or predict the current recovery state; cf. Def. 1.

Parameter(s)	*Value(s)*
LORA rank r	64
LORA α	128
LORA dropout	0.05
Learning rate	0.00095
Batch size	5
Gradient accumulation steps	16
Temperature	0.6
Number of training epochs	2
Quantization	4 bit
PPO [18, Alg. 1]	
Learning rate, # hidden layers	$5148 \cdot 10^{-5}$, 1,
# Neurons/layer	64
# Steps between updates,	2048,
Batch size, discount factor γ	16, 0.99
GAE λ, clip range, entropy coefficient	0.95, 0.2, $2 \cdot 10^{-4}$
Value coefficient, max gradient norm	0.102, 0.5
Feature representation	the original cyborg features [84] & one-hot encoded scan-state & decoy-state for each node

TABLE 7: Hyperparameters.

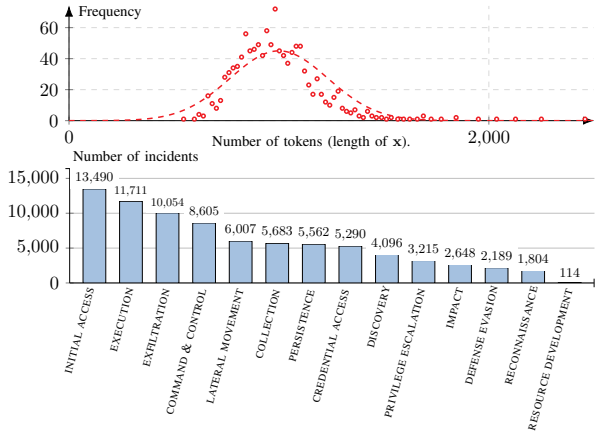


Fig. 14: Distributions of the number of tokens (upper plot) and MITRE ATT&CK TACTICS [77] (lower plot) in our dataset of instruction-answer pairs $(\mathbf{x}, \mathbf{y})$, which we use for fine-tuning the LLM. The dataset is available at [19].

Similarly, each answer either consists of the true recovery state or the optimal response action, both of which are manually selected based on knowledge about the incident.

Since these 500 instruction-answer pairs are too few for effective fine-tuning, we then expand the dataset using synthetic data generated by prompting GEMINI 2.5 [16] and OPENAI O3 [14] with our testbed examples to generate new examples of similar structure but for different types of systems and attacks, yielding a total dataset of size 68,000. This dataset covers a diverse range of attacks, system architectures, and log data. Each instruction-answer pair follows a specific JSON structure. Figure 14 shows the distributions of token counts and MITRE ATT&CK tactics [77] in our dataset. We see that most incidents are described by around 1200 tokens, and the most common attacker tactics are INITIAL ACCESS and EXECUTION. The prompt templates that we use are available at [19].

Our approach of combining testbed examples with synthetic examples is inspired by the studies presented in [85] and [86], which successfully used similar approaches to generate fine-tuning datasets for other domains, e.g., the medical domain.

Type	*Actions*	MITRE ATT&CK technique
Reconnaissance	TCP SYN scan, UDP scan	T1046 service scanning
	TCP XMAS scan	T1046 service scanning
	VULSCAN	T1595 active scanning
	ping-scan	T1018 system discovery
Brute-force	TELNET, SSH	T1110 brute force
	FTP, CASSANDRA	T1110 brute force
	IRC, MONGODB, MYSQL	T1110 brute force
	SMTP, POSTGRES	T1110 brute force
Exploit	CVE-2017-7494	T1210 service exploitation
	CVE-2015-3306	T1210 service exploitation
	CVE-2010-0426	T1068 privilege escalation
	CVE-2015-5602	T1068 privilege escalation
	CVE-2015-1427	T1210 service exploitation
	CVE-2014-6271	T1210 service exploitation
	CVE-2016-10033	T1210 service exploitation
	SQL injection (CWE-89)	T1210 service exploitation

TABLE 8: Attacker actions executed on our testbed to generate the initial examples for our training dataset, which we use for fine-tuning the LLM. Actions are mapped to the corresponding vulnerabilities they exploit, as indicated by the CVE [68] and CWE [87] identifiers. The actions are also linked to the corresponding attack techniques in the MITRE ATT&CK taxonomy [77].

REFERENCES

[1] D. W. Woods, R. Böhme, J. Wolff, and D. Schwarcz, "Lessons lost: Incident response in the age of cyber insurance and breach attorneys," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 2259–2273.

[2] IBM Security and P. Institute, "Cost of a data breach report 2024," IBM, Cambridge, MA, Tech. Rep. 19, 2024, based on breaches at 524 organizations across 17 industries in 16 countries between March 2023 and February 2024.

[3] R. Stevens, D. Votipka, J. Dykstra, F. Tomlinson, E. Quartararo, C. Ahern, and M. L. Mazurek, "How ready is your ready? assessing the usability of incident response playbook frameworks," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, 2022.

[4] D. Schlette, P. Empl, M. Caselli, T. Schreck, and G. Pernul, "Do you play it by the books? a study on incident response playbooks and influencing factors," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 3625–3643.

[5] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.

[6] S. R. Castro, R. Campbell, N. Lau, O. Villalobos, J. Duan, and A. A. Cardenas, "Large language models are autonomous cyber defenders," 2025, https://arxiv.org/abs/2505.04843.

[7] M. Rigaki, O. Lukáš, C. A. Catania, and S. Garcia, "Out of the cage: How stochastic parrots win in cyber security environments," 2023, https://arxiv.org/abs/2308.12086.

[8] H. Mohammadi, J. J. Davis, and M. Kiely, "Leveraging large language models for autonomous cyber defense: Insights from CAGE-2 simulations," *IEEE Intelligent Systems*, pp. 1–8, 2025.

[9] Y. Yan, Y. Zhang, and K. Huang, "Depending on yourself when you should: Mentoring LLM with RL agents to become the master in cybersecurity games," 2024, https://arxiv.org/abs/2403.17674.

[10] S. Hays and J. White, "Employing LLMs for incident response planning and review," 2024, https://arxiv.org/abs/2403.01271.

[11] X. Lin, J. Zhang, G. Deng, T. Liu, X. Liu, C. Yang, T. Zhang, Q. Guo, and R. Chen, "IRCopilot: Automated incident response with large language models," 2025, https://arxiv.org/abs/2505.20945.

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

[13] S. Hussey. (2025, June) Resolve incidents faster with IBM Instana intelligent incident investigation powered by agentic AI. Accessed: 2025-07-08, https://www.ibm.com/new/announcements/resolve-

incidents-faster-with-ibm-instana-intelligent-incident-investigation-powered-by-agentic-ai.
[14] OpenAI, J. Achiam, S. Adler *et al.*, "GPT-4 technical report," 2024, https://arxiv.org/abs/2303.08774.
[15] G. Sriramanan, S. Bharti, V. S. Sadasivan, S. Saha, P. Kattakinda, and S. Feizi, "LLM-check: Investigating detection of hallucinations in large language models," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 34 188–34 216.
[16] G. Comanici, E. Bieber, M. Schaekermann *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," 2025, https://arxiv.org/abs/2507.06261.
[17] G. Team, R. Anil, S. Borgeaud *et al.*, "Gemini: A family of highly capable multimodal models," 2024, https://arxiv.org/abs/2312.11805.
[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017, http://arxiv.org/abs/1707.06347.
[19] K. Hammar, T. Alpcan, and E. C. Lupu, "Supplementary material of the paper "Incident Response Planning Using a Lightweight Large Language Model with Reduced Hallucination"," 2025, code for fine-tuning: https://github.com/Limmen/llm_recovery, code for our testbed: https://github.com/Limmen/csle, dataset: https://huggingface.co/datasets/kimhammar/CSLE-IncidentResponse-V1, video demonstration: https://www.youtube.com/watch?v=e7ckmv5p6cI, fine-tuned LLM and prompts: https://huggingface.co/kimhammar/LLMIncidentResponse.
[20] N. Stakhanova, S. Basu, and J. Wong, "A taxonomy of intrusion response systems," *Int. J. Inf. Comput. Secur.*, vol. 1, no. 1/2, p. 169–184, Jan. 2007.
[21] T. Alpcan and T. Basar, "A game theoretic approach to decision and analysis in network intrusion detection," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 3, 2003, pp. 2595–2600 Vol.3.
[22] A. Applebaum, S. Johnson, M. Limiero, and M. Smith, "Playbook oriented cyber response," in *2018 National Cyber Summit (NCS)*, 2018, pp. 8–15.
[23] Splunk, "Automate incident response with playbooks and actions in Splunk mission control," 2025, https://help.splunk.com/en/splunk-enterprise-security-7/mission-control/investigate-and-respond-to-threats/automate-incident-response/automate-incident-response-with-playbooks-and-actions-in-splunk-mission-control.
[24] CISA, "Cybersecurity incident & vulnerability response playbooks," 2021, https://www.cisa.gov/resources-tools/resources/federal-government-cybersecurity-incident-and-vulnerability-response-playbooks.
[25] OASIS, "Cacao security playbooks version 2.0," 2023, https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.html.
[26] K. Hammar and R. Stadler, "Intrusion tolerance for networked systems through two-level feedback control," in *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2024, pp. 338–352.
[27] A. V. Singh, E. Rathbun, E. Graham, L. Oakley, S. Boboila, A. Oprea, and P. Chin, "Hierarchical multi-agent reinforcement learning for cyber network defense," 2024, https://arxiv.org/abs/2410.17351.
[28] K. Hammar, T. Li, R. Stadler, and Q. Zhu, "Adaptive security response strategies through conjectural online learning," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 4055–4070, 2025.
[29] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "RRE: A game-theoretic intrusion response and recovery engine," in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009, pp. 439–448.
[30] T. Alpcan and T. Basar, *Network Security: A Decision and Game-Theoretic Approach*, 1st ed. USA: Cambridge University Press, 2010.
[31] CAGE, "TTCP CAGE challenge 2," in *AAAI-22 Workshop on Artificial Intelligence for Cyber Security (AICS)*, 2022, https://github.com/cage-challenge/cage-challenge-2.
[32] K. Hammar, Y. Li, T. Alpcan, E. C. Lupu, and D. Bertsekas, "Adaptive network security policies via belief aggregation and rollout," 2025, https://arxiv.org/abs/2507.15163.
[33] S. Vyas, J. Hannay, A. Bolton, and P. P. Burnap, "Automated cyber defence: A review," 2023, code: https://github.com/john-cardiff/-cyborg-cage-2.
[34] K. Hammar, N. Dhir, and R. Stadler, "Optimal defender strategies for CAGE-2 using causal modeling and tree search," 2024, https://arxiv.org/abs/2407.11070.
[35] T. Li, K. Hammar, R. Stadler, and Q. Zhu, "Conjectural online learning with first-order beliefs in asymmetric information stochastic games," in *2024 IEEE 63rd Conference on Decision and Control (CDC)*, 2024, pp. 6780–6785.
[36] A. Applebaum, C. Dennler, P. Dwyer, M. Moskowitz, H. Nguyen, N. Nichols, N. Park, P. Rachwalski, F. Rau, A. Webster, and M. Wolk, "Bridging automated to autonomous cyber defense: Foundational analysis of tabular Q-learning," in *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, 2022.
[37] A. Ramamurthy and N. Dhir, "General autonomous cybersecurity defense: Learning robust policies for dynamic topologies and diverse attackers," 2025, https://arxiv.org/abs/2506.22706.
[38] Z. Huang, J. Robin, N. Herbaut, N. B. Rabah, and B. L. Grand, "Toward an intent-based and ontology-driven autonomic security response in security orchestration automation and response," 2025, https://arxiv.org/abs/2507.12061.
[39] A. Shaked, Y. Cherdantseva, and P. Burnap, "Model-based incident response playbooks," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ser. ARES '22. New York, NY, USA: Association for Computing Machinery, 2022.
[40] M. Akbari Gurabi, L. Nitz, A. Bregar, J. Popanda, C. Siemers, R. Matzutt, and A. Mandal, "Requirements for playbook-assisted cyber incident response, reporting and automation," *Digital Threats*, vol. 5, no. 3, Oct. 2024.
[41] G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, "PentestGPT: Evaluating and harnessing large language models for automated penetration testing," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 847–864.
[42] M. Rodriguez, R. A. Popa, F. Flynn, L. Liang, A. Dafoe, and A. Wang, "A framework for evaluating emerging cyberattack capabilities of AI," 2025, https://arxiv.org/abs/2503.11917.
[43] J. Deng, X. Li, Y. Chen, Y. Bai, H. Weng, Y. Liu, T. Wei, and W. Xu, "RACONTEUR: A knowledgeable, insightful, and portable LLM-powered shell command explainer," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.
[44] A. Stafeev, T. Recktenwald, G. D. Stefano, S. Khodayari, and G. Pellegrino, "Yurascanner: Leveraging LLMs for task-driven web app scanning," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.
[45] P. Liu, J. Liu, L. Fu, K. Lu, Y. Xia, X. Zhang, W. Chen, H. Weng, S. Ji, and W. Wang, "Exploring ChatGPT's capabilities on vulnerability management," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 811–828.
[46] M. Allamanis, M. Arjovsky, C. Blundell *et al.*, "From naptime to big sleep: Using large language models to catch vulnerabilities in real-world code," 2024, https://googleprojectzero.blogspot.com/2024/10/from-naptime-to-big-sleep.html.
[47] Y. Liu, Y. Xue, D. Wu, Y. Sun, Y. Li, M. Shi, and Y. Liu, "PropertyGPT: LLM-driven formal verification of smart contracts through retrieval-augmented property generation," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.
[48] V. Gohil, M. DeLorenzo, V. V. A. S. V. Nallam, J. See, and J. Rajendran, "LLMPirate: LLMs for black-box hardware IP piracy," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.
[49] Y. Yang, J. Liu, K. Chen, and M. Lin, "The midas touch: Triggering the capability of LLMs for RM-API misuse detection," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.
[50] C. S. Xia, M. Paltenghi, J. Le Tian, M. Pradel, and L. Zhang, "Fuzz4all: Universal fuzzing with large language models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024.
[51] X. Ma, L. Luo, and Q. Zeng, "From one thousand pages of specification to unveiling hidden bugs: Large language model assisted fuzzing of matter IoT devices," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 4783–4800.
[52] J. Liu, Y. Yang, K. Chen, and M. Lin, "Generating API parameter security rules with LLM for API misuse detection," in *32nd Annual*

Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025. The Internet Society, 2025.

[53] D. Wu, X. Wang, Y. Qiao, Z. Wang, J. Jiang, S. Cui, and F. Wang, "NetLLM: Adapting large language models for networking," in *Proceedings of the ACM SIGCOMM 2024 Conference*, ser. ACM SIGCOMM '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 661–678.

[54] M. Arazzi, D. R. Arikkat, S. Nicolazzo, A. Nocera, R. Rehiman K.A., V. P., and M. Conti, "NLP-based techniques for cyber threat intelligence," *Computer Science Review*, vol. 58, p. 100765, 2025.

[55] P. Hu, R. Liang, and K. Chen, "Degpt: Optimizing decompiler output with LLM," in *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024.

[56] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 1st ed. USA: John Wiley & Sons, Inc., 2001.

[57] A. A. Ganin, E. Massaro, A. Gutfraind, N. Steen, J. M. Keisler, A. Kott, R. Mangoubi, and I. Linkov, "Operational resilience: concepts, design and analysis," *Scientific Reports*, vol. 6, no. 1, p. 19540, Jan 2016.

[58] L. Li, X. Zhang, X. Zhao, H. Zhang, Y. Kang, P. Zhao, B. Qiao, S. He, P. Lee, J. Sun, F. Gao, L. Yang, Q. Lin, S. Rajmohan, Z. Xu, and D. Zhang, "Fighting the fog of war: Automated incident detection for cloud systems," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, Jul. 2021, pp. 131–146.

[59] A. Morse, "Investigation: Wannacry cyber attack and the NHS," 2017, national Audit Office UK.

[60] S. M. T. I. Tonmoy, S. M. M. Zaman, V. Jain, A. Rani, V. Rawte, A. Chadha, and A. Das, "A comprehensive survey of hallucination mitigation techniques in large language models," 2024, https://arxiv.org/abs/2401.01313?

[61] DeepSeek-AI, D. Guo, D. Yang *et al.*, "DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," 2025, https://arxiv.org/abs/2501.12948.

[62] O. Ayala and P. Bechard, "Reducing hallucination in structured outputs via retrieval-augmented generation," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, Y. Yang, A. Davani, A. Sil, and A. Kumar, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 228–238.

[63] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[64] X. Chen, R. Aksitov, U. Alon, J. Ren, K. Xiao, P. Yin, S. Prakash, C. Sutton, X. Wang, and D. Zhou, "Universal self-consistency for large language model generation," 2023, https://arxiv.org/abs/2311.17311.

[65] Y. Weng, M. Zhu, F. Xia, B. Li, S. He, S. Liu, B. Sun, K. Liu, and J. Zhao, "Large language models are better reasoners with self-verification," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2550–2575.

[66] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-refine: iterative refinement with self-feedback," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2023.

[67] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[68] MITRE, "CVE database," 2022, https://cve.mitre.org/.

[69] P. E. Kaloroumakis and M. J. Smith, "Toward a knowledge graph of cybersecurity countermeasures," The MITRE Corporation, Annapolis Junction, MD, Technical Report, 2021, approved for Public Release; Distribution Unlimited.

[70] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.

[71] D. Y. Huang, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A. C. Snoeren, and D. McCoy, "Tracking ransomware end-to-end," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 618–631.

[72] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA '99. USA: USENIX Association, 1999, p. 229–238.

[73] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "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*, INSTICC. SciTePress, 2018, pp. 108–116.

[74] M. Landauer, F. Skopik, and M. Wurzenberger, "Introducing a new alert data set for multi-step attack analysis," in *Proceedings of the 17th Cyber Security Experimentation and Test Workshop*, ser. CSET '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 41–53.

[75] Wazuh Inc, "Wazuh - the open source security platform," 2022.

[76] AT&T Cybersecurity, "AlienVault Open Threat Exchange (OTX)," https://otx.alienvault.com, 2021, https://otx.alienvault.com.

[77] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre ATT&CK: Design and philosophy," in *Technical report*. MITRE, 2018.

[78] M. Landauer, F. Skopik, M. Frank, W. Hotwagner, M. Wurzenberger, and A. Rauber, "Maintainable log datasets for evaluation of intrusion detection systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 3466–3482, 2023.

[79] K. Hammar and R. Stadler, "Intrusion prevention through optimal stopping," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2333–2348, 2022.

[80] K. Hammar, "Optimal security response to network intrusions in it systems," Ph.D. dissertation, KTH Royal Institute of Technology, 2024.

[81] Y. A. Yadkori, I. Kuzborskij, D. Stutz, A. György, A. Fisch, A. Doucet, I. Beloshapka, W.-H. Weng, Y.-Y. Yang, C. Szepesvári, A. T. Cemgil, and N. Tomasev, "Mitigating LLM hallucinations via conformal abstention," 2024, https://arxiv.org/abs/2405.01563.

[82] D. Bertsekas, *Rollout, Policy Iteration, and Distributed Reinforcement Learning*, ser. Athena scientific optimization and computation series. Athena Scientific, 2021.

[83] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*, 3rd ed. Athena Scientific, 2007.

[84] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, "Cyborg: A gym for the development of autonomous cyber agents," *CoRR*, 2021, https://arxiv.org/abs/2108.09118.

[85] Y. Wang, H. Ivison, P. Dasigi, J. Hessel, T. Khot, K. R. Chandu, D. Wadden, K. MacMillan, N. A. Smith, I. Beltagy, and H. Hajishirzi, "How far can camels go? exploring the state of instruction tuning on open resources," 2023, https://arxiv.org/abs/2306.04751.

[86] H. Yu, T. Cheng, Y. Cheng, and R. Feng, "FineMedLM-o1: Enhancing the medical reasoning ability of LLM from supervised fine-tuning to test-time training," 2025, https://arxiv.org/abs/2501.09213.

[87] MITRE, "CWE list," 2023, https://cwe.mitre.org/index.html.