IRCopilot: Automated Incident Response with Large Language Models

Xihuan Lin
Fujian Agriculture and Forestry University
Fuzhou, China
xihuan.lin@fafu.edu.cn

Jie Zhang
IHPC and CFAR, A*STAR
Singapore, Singapore
zhang_jie@cfar.a-star.edu.sg

Gelei Deng
Nanyang Technological University
Singapore, Singapore
gelei.deng@ntu.edu.sg

Tianzhe Liu
Fujian Police College
Fuzhou, China
tianzheliu@stu.fjpsc.edu.cn

Xiaolong Liu
Fujian Agriculture and Forestry University
Fuzhou, China
xlliu@fafu.edu.cn

Changcai Yang
Fujian Agriculture and Forestry University
Fuzhou, China
changcaiyang@gmail.com

Tianwei Zhang
Nanyang Technological University
Singapore, Singapore
tianwei.zhang@ntu.edu.sg

Qing Guo*
IHPC and CFAR, A*STAR
Singapore, Singapore
guo_qing@cfar.a-star.edu.sg

Riqing Chen*
Fujian Agriculture and Forestry University
Fuzhou, China
riqing.chen@fafu.edu.cn

ABSTRACT

Incident response plays a pivotal role in mitigating the impact of cyber attacks. In recent years, the intensity and complexity of global cyber threats have grown significantly, making it increasingly challenging for traditional threat detection and incident response methods to operate effectively in complex network environments. While Large Language Models (LLMs) have shown great potential in early threat detection, their capabilities remain limited when it comes to automated incident response after an intrusion. To address this gap, we construct an incremental benchmark based on real-world incident response tasks to thoroughly evaluate the performance of LLMs in this domain. Our analysis reveals several key challenges that hinder the practical application of contemporary LLMs, including context loss, hallucinations, privacy protection concerns, and their limited ability to provide accurate, context-specific recommendations. In response to these challenges, we propose IRCopilot, a novel framework for automated incident response powered by LLMs. IRCopilot mimics the three dynamic phases of a real-world incident response team using four collaborative LLM-based session components. These components are designed with clear divisions of responsibility, reducing issues such as hallucinations and context loss. Our method leverages diverse prompt designs and strategic responsibility segmentation, significantly improving the system's practicality and efficiency. Experimental results demonstrate that IRCopilot outperforms baseline LLMs across key benchmarks, achieving sub-task completion rates of 150%, 138%, 136%, 119%, and 114% for various response tasks. Moreover, IRCopilot exhibits robust performance on public incident response platforms and in real-world attack scenarios, showcasing its strong applicability.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**.

KEYWORDS

Large Language Models; Incident Response; AI for Security

ACM Reference Format:
Xihuan Lin, Jie Zhang, Gelei Deng, Tianzhe Liu, Xiaolong Liu, Changcai Yang, Tianwei Zhang, Qing Guo, and Riqing Chen. 2025. IRCopilot: Automated Incident Response with Large Language Models. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Incident Response (IR) is a cornerstone of system and network security, offering effective countermeasures to mitigate potential losses caused by cyber attacks. As shown in Figure 1, the lifecycle of IR comprises three key stages: detection, response, and recovery, which play critical roles in the broader information security framework [41]. Achieving efficient incident response requires timely decision-making, cross-functional collaboration, and the ability to adapt swiftly to evolving threats. However, current IR practices heavily rely on manual analysis, specialized expertise, structured processes and collaboration, making them increasingly inadequate in addressing today's complex and frequent cyber threats.

*Qing Guo and Riqing Chen are the corresponding authors

CCS '25, October 13–17, 2025, Taipei, Taiwan

ACM ISBN 978-1-4503-XXXX-X/2018/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

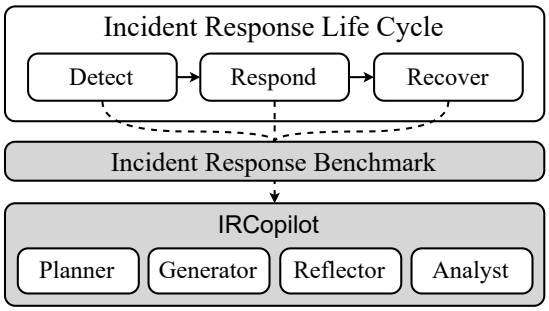


Figure 1: Two main contributions of this work: 1) We build an Incident Response (IR) benchmark to evaluate the contemporary LLMs on IR tasks, in terms of three stages. 2) We propose IRCopilot to enhance the performance on IR tasks.

To gain a clearer understanding of the current IR landscape, we conducted a preliminary survey. Our findings highlight several urgent concerns: (1) Continuous expansion of the attack surface and increase in vulnerability. (2) Diversification of vulnerability types and cybersecurity incidents. (3) Prominence of industry and scenario specificity. . (4) Rising technical, economic, and ethical challenges from cyber attack losses. Traditional security operations and response models, which depend heavily on manual efforts, are no longer sufficient to address these realities. Consequently, enhancing automation in IR has becomes a critical research objective.

In recent years, the development of Natural Language Processing (NLP) [28] and Large Language Models (LLMs), such as GPT [23, 26] and BERT [30], has presented new opportunities for automated analysis and decision-making. In the field of cybersecurity, LLMs have demonstrated strong potential [29, 34, 48] and are expected to serve as "intelligent assistants" in IR. Some preliminary studies have utilized LLMs to assist in security analysis and threat intelligence extraction [24, 38, 44], but they primarily focus on early stages of threat scenarios (such as intelligence gathering and detection), lacking comprehensive response (analysis) and recovery to threats. Fully leveraging the language understanding and multitask inference capabilities of LLMs for automated IR remains an area that has not been thoroughly explored.

To bridge these gaps, there is an urgent need of a comprehensive benchmark to evaluate the capabilities of LLMs in incident response. Currently, evaluation frameworks designed for specific tasks are lacking. To bridge these gaps, we develop an IR benchmark encompassing three mainstream platforms (*i.e.*, XuanJi [21], ZGSF [22], and TryHackMe [19]), comprising a total of 12 case machines and 130 sub-tasks. This benchmark covers the three main stages of the IR lifecycle as defined by NIST 3.0 [41], with each sub-task mapped to more detailed execution steps within the corresponding guidelines. By assessing the completion rates of sub-tasks, we can comprehensively characterize the model's capabilities in IR.

We then conduct an exploratory study using the designed benchmark to evaluate the capabilities of LLMs in IR tasks, with a focus on handling real-world complexities. We test four commercial LLMs (*i.e.*, GPT-4 [4], GPT-4o [4], Claude-3.5-Sonnet [3], and GPT-o1 [4]) and one open-source LLM (DeepSeek-V3 [9]) under controlled conditions with identical environment configurations and prompts. Experiment results show that LLMs cannot effectively overcome these tasks. Specifically, they face the following challenges in IR tasks: (1) difficulty in formulating appropriate response strategies; (2) generation of inaccurate guidance or commands; (3) overlooking critical details; (4) limited long-term memory and poor handling of multiple lines of evidence; (5) potential privacy risks due to exposure to sensitive information.

To enhance automation in incident response, minimize human errors, shorten response time, and provide intelligent decision support during unexpected security incidents, we propose IRCopilot, an interactive automated IR system driven by LLMs. Targeting the identified challenges, IRCopilot consists of four functional components: Planner, Generator, Reflector, and Analyst (see Figure 1), and can be divided into three stages of thought: reasoning, action, and reflection. (1) In the reasoning stage, the Planner is responsible for overall task planning and decomposing complex problems into more detailed sub-tasks. The Analyst evaluates and analyzes the execution results and feeds the analysis conclusions back to the Planner to support subsequent decision-making and optimization. Such decision process effectively overcome the challenges of LLMs to zero-shot the correct response sequence. (2) In the action stage, the Generator produces specific execution plans based on the Planner's instructions. Since the overall task is decomposed to sub operations individually, the attention/memeory issues of LLMs [55, 57] are effectively mitigated. (3) Finally, in the reflection stage, the Reflector integrates multi-source information to review the execution process and propose improvement suggestions, forming a complete closed loop from strategy formulation to continuous optimization. By integrating the aforementioned four LLM components, IRCopilot can efficiently support various functions such as incident detection and classification, evidence analysis and threat intelligence correlation, emergency plan formulation, interactive decision-making and execution recommendations, as well as built-in security compliance auditing and data protection. Throughout this process, a seamless connection is achieved from high-level strategy to precise execution and intelligent data analysis, ensuring the coherence and efficiency of IR.

We evaluate IRCopilot across various testing scenarios to verify its effectiveness and scope of applicability. The results indicate that, in the benchmark tests, IRCopilot achieve sub-task completion rates of 1.5×, 1.38×, 1.36×, 1.19×, and 1.14× higher than directly applying GPT-4, DeepSeek-V3, GPT-4o, Claude-3.5-Sonnet, and GPT-o1 respectively, thus demonstrating significantly better performance than the baseline methods. In real-world challenges (*i.e.*, TryHackMe, XuanJi, and ZGSF), IRCopilot successfully resolve the majority of incident response tasks, underscoring its practical value in enhancing efficiency and accuracy. Furthermore, we conduct an evaluation on five real network attack cases and find that IRCopilot effectively identifies and extracts critical traces and information left by attackers. More details are available on Anonymous GitHub [12].

To summarize, we make the following key contributions:

- **Comprehensive Survey of Recent Incident Response Trends and Cybersecurity Landscape.** This survey reveals numerous findings they are currently encountering and highlights the limitations of existing solutions.
- **Development of a Comprehensive Incident Response Benchmark.** Our benchmark covers numerous target machines from

prominent platforms (TryHackMe, XuanJi, and ZGSF). It comprises 12 representative targets (with 130 sub-tasks) and spans the three key stages of incident response, partially drawing on real-world attack scenarios to provide fair and holistic evaluations, and highlights the challenges faced by LLMs in IR tasks. As far as we know, this is the first comprehensive benchmark in this domain dedicated to specific incident response workflows, with stage-wise evaluation and comparison capabilities.

- **Design of an Innovative LLM-Based Incident Response System.** Our automated and interactive system, IRCopilot, provides efficient and innovative incident handling capabilities. Through various technological innovations, it significantly boosts incident response accuracy and execution efficiency while minimizing its impact on the system, ensuring overall stability and practicality.
- **Systematic Evaluation and Real-World Validation.** The evaluation results indicate that incident response analysis based on the GPT-4o series successfully achieve objectives in the vast majority of tasks, thus demonstrating the practicality and reliability of IRCopilot in complex environments. Furthermore, we validate its performance in real-world scenarios, encompassing two major operating systems and various attack methods. The findings show that this approach is versatile and can be effectively extended to real-world incident response.

2 BACKGROUND & RELATED WORK

2.1 Global Cybersecurity Landscape

In recent years, the global cybersecurity ecosystem has been facing unprecedented challenges of complexity and dynamism. We conducted an in-depth analysis of over 200 annual reports on cybersecurity or IR, and draw the following observations.

(1) Continuous Expansion of Attack Surface and Increase in Vulnerability. The increasing complexity and high interconnectivity of software systems are continuously expanding the potential attack surface, driving a sustained rise in the number and severity of vulnerabilities. According to publicly available data from the NVD (National Vulnerability Database) [17] (showing in Appendix Figure 8a), the number of published CVEs (Common Vulnerabilities and Exposures) has surged dramatically from approximately 7,928 in 2014 to 40,287 in 2024, exhibiting an exponential upward trend. Additionally, the cumulative number of CVEs has grown from less than 70,000 to over 262,000 within a decade. This means that the amount of vulnerability intelligence data that security teams need to handle is massive.

(2) Diversification of Vulnerability Types and Cybersecurity Incidents. These vulnerabilities are not confined to a single exploitation technique but exhibit highly diversified and specialized characteristics (Appendix Figure 8b). Between 2014 and 2023, the number and proportion of vulnerabilities such as code execution, privilege escalation, denial of service, and information disclosure have all increased. A significant point of concern is the increase in remote code execution and privilege escalation vulnerabilities, as these types of vulnerabilities are often directly exploited by attackers to penetrate core system components, ultimately enabling full control over an organization's internal resources.

(3) Prominence of Industry and Scenario Specificity. Meanwhile, the threats faced by organizations across different sectors and industries have become increasingly differentiated and multifaceted. According to data released by Check Point[6], the average weekly number of attacks in critical infrastructure sectors such as finance, government, education, and healthcare has increased significantly between 2020 and 2023 (Appendix Figure 7). This indicates that customized attack patterns targeting different industries are becoming increasingly sophisticated.

(4) Comprehensive Upgrade of Cybersecurity Requirements. In addition, empirical studies and data analysis in the field of IR also reveal a complex landscape of evolving threats. According to data released by Unit 42 [20], between 2021 and 2023, ransomware attacks (including encrypted and unencrypted ransomware), BEC (Business Email Compromise), network intrusions and Web application breaches have become the main types of investigations and responses (Appendix Figure 10). The variations in attack techniques and focus areas across different years reflect the flexibility and dynamic adjustments in attackers' strategies. This increases not only the workload and cognitive load of IR teams, but also the risks of misjudgments, delays, and resource wastage. Correspondingly, Kaspersky [13] (Appendix Figure 9) presents statistics on security incidents and risks triggered by various initial attack vectors. The results indicate that these attacks may lead not only to direct economic losses, but also to data breaches, service interruptions, and privacy and ethical issues.

2.2 Incident Response

Cybersecurity incident response involves organized, systematic actions taken by an organization upon detecting a security incident or potential breach due to unauthorized activities. These activities may compromise the confidentiality, integrity, or availability of systems or assets, or violate laws, policies, or security protocols. The aim is to reduce harm, limit losses, and quickly restore normal operations. As shown in Figure 1, the lifecycle of IR consists of three key stages [41]: Detect, Respond and Recover. In the Detect phase, potential cybersecurity threats and compromises are identified and analyzed. The Respond phase analyzes the system and takes necessary measures to address detected cybersecurity incidents. Finally, in the Recover phase, efforts are made to restore affected assets and operations to their normal state.

Limitations. The increasing persistence, stealth, and unpredictability of cyber threats challenge traditional threat detection methods. Conventional rule-based and machine learning methodologies excel in a priori tasks, such as real-time intrusion detection and threat analysis, due to their low latency and precision [54]. However, these approaches stumble when it comes to incident response, particularly in post-incident forensic analysis and adaptive mitigation. These methods rely on fixed rules and pre-trained models, which hampers their ability to keep up with an ever-changing threat landscape, leaving them less suited to the dynamic, reactive demands of IR. In the past, IR has relied heavily on manual efforts and expert knowledge, with teams laboriously investigating the overall scope of incidents, analyzing threats, and addressing critical issues by hand. But as threats become more sophisticated, manual approach struggles to deliver timely and efficient outcomes. Although traditional a priori methods can help with initial assessments, their task-specific focus and limited scalability fall short of the broader demands of

IR. Consequently, integrating artificial intelligence to automate IR remains a significant challenge. Moreover, cyber threats do not just exploit system vulnerabilities, they can also stem from diverse sources like internal actors and espionage, making analysis even trickier. When faced with massive amounts of network data and unpredictable scenarios, security researchers require more adaptable tools. LLMs show promise here, leveraging their flexibility and contextual understanding [27].

2.3 Large Language Models and Their Applications for Cybersecurity

With the development of LLMs recently, significant transformations have occurred across various fields. Renowned models such as GPT [23, 26], Llama [31], Claude [7], and DeepSeek [8] have demonstrated excellent performance in numerous domains, including text generation [37], translation [50], programming [36], and summarization [56].

To further enhance the performance of LLMs in complex task scenarios, researchers have introduced and adopted various cutting-edge planning and reasoning methods, collectively known as prompt engineering. These methods include CoT (Chain of Thought) [51], ToT (Tree of Thoughts) [52], RAG (Retrieval Augmented Generation) [35], ReAct (Reasoning and Acting) [53], and Reflexion [47]. They enable LLMs to generate more factually grounded responses and perform better in intricate analytical tasks. For instance, ToT constructs a thought tree structure, prompting the model to engage in multi-level reasoning before providing the final answer, thereby improving the accuracy and reliability of responses. These characteristics make ToT particularly suitable for the deep analysis and decision-making required in automated cybersecurity IR. We will investigate how to employ these prompt engineering techniques to optimize the application effectiveness of LLMs in automated IR.

LLMs, with their deep contextual understanding and reasoning capabilities, have demonstrated considerable potential in numerous areas of cybersecurity, including code analysis [42], penetration testing [29, 34], OSINT (Open Source Intelligence) [46], vulnerability remediation and management [39, 43], malicious webpage identification [38], attack pattern extraction from CTI [24], log-based anomaly detection [44], and developing privacy-preserving models for IoT [32]. However, existing research focuses mainly on early threat stages, such as intelligence gathering and detection. With the escalating severity of cyber threats, the domain of automated IR post-intrusion remains relatively underdeveloped. To address this gap, we propose an LLM-based automated incident response framework. Furthermore, there is currently a lack of a benchmark capable of evaluating the performance of LLMs in IR. Our research will establish a professional, objective, and representative benchmark to provide a reliable reference for subsequent studies in this field.

3 THREAT MODEL

We consider a practical scenario with three parties: a victim, an attacker, and a third-party IR service provider (responder).

Victim's Capabilities & Goals. Operating from a white-box perspective, the victim's IT infrastructure is equipped with an intrusion detection system (IDS) capable of identifying unauthorized access or anomalous network activity. Upon detecting a potential threat, the victim promptly informs the responder. Their goal is to minimize attack impact while preserving critical business operations.

Attacker's Capabilities & Goals. From a gray-box perspective, the attacker seeks to steal sensitive data, extort profit, or disrupt operations by gaining unauthorized access. Their capabilities include: (1) adequate resources (e.g., computing power, libraries) and experience in multi-stage attacks and evasion techniques. (2) partial knowledge of the victim's security defenses, allowing bypass of specific protections. Furthermore, they (3) inevitably leave traces in the compromised system.

Responder's Capabilities & Goals. The responder operates from a gray-box perspective, possessing: (1) limited privileges to access victim data, restricted to information necessary for IR. (2) partial knowledge of attacker traces and compromised assets. Their responsibility to respond effectively while minimizing disruptions to the victim's core business operations.

4 A BENCHMARK FOR INCIDENT RESPONSE

4.1 Motivation

The evaluation of LLMs in the context of IR necessitates an objective and comprehensive benchmark. Existing benchmarks in this domain [1, 2, 33], while valuable, exhibit significant limitations when applied to LLM assessment. These benchmarks primarily emphasize the completion of overarching task workflows, often neglecting the detailed evaluation of specific sub-tasks and intermediate steps within the IR process. Although such macro-oriented benchmarks help determine the success or failure of the final response, they fail to adequately reflect the progressive advancements and value accumulation at each stage of the IR. Consequently, these benchmarks struggle to accurately measure the comprehensive capabilities of LLMs in the details and steps of IR, and they cannot provide strong support for precise performance evaluation and optimization improvements of the models.

To address the above issues, we propose a comprehensive incident response benchmark that meets the following criteria:

Comprehensive and Realistic Tasks. Benchmark designs should incorporate a diverse range of IR task types to ensure strong relevance to real-world scenarios. Beyond supporting mainstream operating systems, it is essential to include contexts based on actual cybersecurity incidents.

Difficulty Gradient Design. To ensure broad applicability, the benchmark should feature layered difficulty levels. This strategy not only enables a more precise evaluation of IR personnel's capabilities at different levels, but also provides a flexible framework for adaptation and serves for future research and practice.

Phase-oriented Effectiveness Evaluation. To assess both the final results and key stages of IR, the benchmark should adopt a multi-phase evaluation system. This enables a thorough analysis of the entire IR. Using measurable indicators, it clarifies the impact of each stage on response effectiveness and efficiency, offering precise guidance for strategy optimization and capability improvement.

4.2 Benchmark Design

Based on the aforementioned criteria, we develop a new IR benchmark to systematically map tasks in real-world settings. We invite

Table 1: Performance of contemporary LLMs on the benchmark.

	Overall				Sub-task			
LLMs	Easy (3)	Medium (6)	Hard (3)	Total (12)	Easy (36)	Medium (62)	Hard (32)	Total (130)
GPT-4	0 (0.00%)	1 (16.67%)	0 (0.00%)	1 (8.33%)	22 (61.11%)	39 (62.90%)	15 (46.88%)	**76 (58.46%)**
DeepSeek-V3	0 (0.00%)	1 (16.67%)	0 (0.00%)	1 (8.33%)	20 (55.56%)	43 (69.35%)	22 (68.75%)	**85 (65.38%)**
GPT-4o	1 (33.33%)	1 (16.67%)	1 (33.33%)	3 (25.00%)	28 (77.78%)	45 (72.58%)	21 (65.63%)	**94 (72.31%)**
Claude-3.5-Sonnet	2 (66.67%)	2 (33.33%)	1 (33.33%)	5 (41.67%)	31 (86.11%)	50 (80.65%)	27 (84.38%)	**108 (83.08%)**
GPT-o1	1 (33.33%)	3 (50.00%)	1 (33.33%)	5 (41.67%)	31 (86.11%)	54 (87.10%)	28 (87.50%)	**113 (86.92%)**
Average	**0.8 (26.67%)**	**1.6 (26.67%)**	**0.6 (20.00%)**	**3 (25.00%)**	**26.4 (73.33%)**	**46.2 (74.52%)**	**22.6 (70.63%)**	**95.2 (73.23%)**

Table 2: Causes of LLM failures in incident response.

Failure Reasons	GPT-4	DeepSeek	GPT-4o	Claude	GPT-o1	Total
False IR Strategy	20	20	14	12	8	**74**
False Command Generation	11	6	10	2	1	**30**
Key Information Ignored	7	7	5	1	4	**24**
False Guidance Generation	8	4	2	2	4	**20**
False Result Interpretation	5	5	2	2	0	**14**
Session context lost	3	3	3	3	0	**12**
Total	54	45	36	22	17	**174**

three seasoned anonymous cybersecurity experts to contribute to the design of the benchmark. Each expert brings more than five years of experience in incident response and vulnerability analysis, holds a CISSP [5] certification, and has documented multiple CVE vulnerability discoveries. The design and development process involves several key stages.

Task Selection. In the task selection process, we focus on constructing simulated environments based on real cybersecurity incident clues. First, we collect IR tasks from mainstream cybersecurity practice platforms (*i.e.*, XuanJi [21], ZGSF [22] and TryHackMe [19]). Our selection criteria aim to encompass as broadly as possible the common types of incidents encountered in reality. To ensure comprehensive and high-quality tasks, we rigorously review all IR-related tasks and virtual environments across each platform. From this, we select a representative subset of tasks covering various incident types and response methods. To simulate real-world IR scenarios of varying complexity, experts categorize these tasks into three levels—Easy, Medium, and Hard—based on manual testing, discussion, and task completion rates. Importantly, we prioritize the selection of tasks that are less likely to have been included in LLM training corpora. In particular, we focus on tasks published after common model training cutoffs and avoid widely publicized or overly replicated content. This helps mitigate potential data leakage and ensures fair evaluation of LLM generalization capabilities. Further details are discussed in Sec. 7. Additionally, all selected tasks exclude real-world privacy or sensitive data, thereby avoiding potential ethical and legal concerns.

Task Decomposition. To ensure the rigor and systematic nature of our research methodology, we break down each IR task into a series of sub-tasks across three steps: clue acquisition, information analysis, and incident handling and recovery. The entire process strictly adheres to the guiding principles of NIST SP 800-61 Rev. 3 [41], mapping each sub-task to the specific step breakdowns defined in the guidelines. Ultimately, we develop a comprehensive list of sub-tasks for each benchmark task, which is provided in Appendix Table 10.

Benchmark Validation. To ensure the scientific validity and effectiveness of the benchmark, we implement a rigorous evaluation and iterative optimization process. Specifically, the experts independently complete the assigned benchmark tasks and document a detailed IR process based on their performance and results. Based on the experts' feedback, we selectively adjust the task decomposition strategy, focusing on addressing potential multiple response paths within the tasks, thereby enhancing the applicability and flexibility of the benchmarks in practical applications.

Through the process outlined above, we develop a comprehensive benchmark that systematically reflects real-world IR strategies. This benchmark includes 12 IR objectives, each assigned varying difficulty levels, and is further subdivided into 130 specific sub-tasks across 27 categories. These objectives and sub-tasks fully represent the essential technologies, skills, and scenarios integral to IR. Designed with careful attention to task quantity and scope, the benchmark accommodates diverse real-world applications while effectively evaluating multi-level IR capabilities. Detailed content and classifications are published on our Anonymous Github [12] for researchers and practitioners to reference. Furthermore, publicly available high-difficulty cases are currently limited, and we will continuously expand the benchmark to enhance its adaptability.

4.3 Evaluation on Contemporary LLMs

We launch a preliminary exploration to evaluate LLMs' capabilities in IR tasks on the benchmark constructed above, using a human-in-the-loop strategy [11, 25, 29, 49]. The human expert acts solely as an executor, adhering to the LLM's instructions without offering independent insights or decisions. We assess the performance of three state-of-the-art commercial LLMs (GPT-4, GPT-4o, and Claude-3.5-Sonnet), along with one open-source LLM (DeepSeek-V3) and the reasoning-focused LLM (GPT-o1). To ensure fair comparisons, all experiments are performed under consistent conditions, employing identical prompts and environmental settings. The LLMs are explicitly instructed not to use external automated tools as the purpose is to evaluate LLMs' inherent knowledge and capabilities. An incident response consists of multiple interconnected tasks spanning the detection, response, and recovery phases. *Success is only achieve when all sub-tasks in these phases are completed successfully.*

Table 1 illustrates the performance of the evaluated LLMs across tasks of varying difficulty levels. Among the models, Claude-3.5-Sonnet achieve the highest overall performance, successfully completing 5 out of 12 tasks, with a sub-task completion rate of 83.08%. In contrast, GPT-4o, DeepSeek-V3, and GPT-4 achieve lower rates of 72.31%, 65.38%, and 58.46%, respectively. Across all models, LLMs

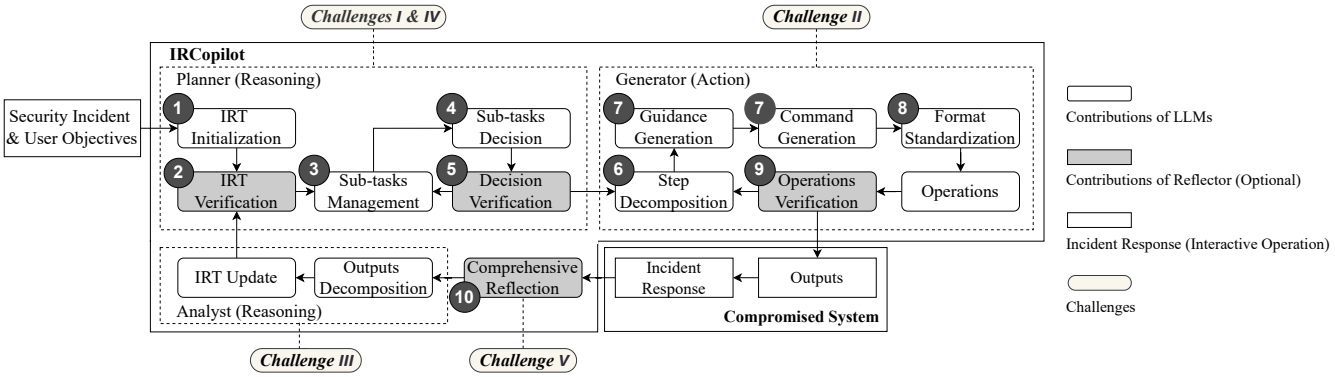


Figure 2: Workflow of IRCopilot. This figure illustrates the principle of IRCopilot, structured into three cognitive stages: Reasoning, Action, and Reflection. 1. In the Reasoning phase, we design the Planner and Analyst to maintain the IRT and tackle *Challenges I, III, and IV* mentioned in Sec. 4.3 through step-by-step reasoning. 2. In the Action phase, the Generator distributes and generates guidance and commands to mitigate *Challenge II*. 3. Finally, in the Reflection phase, the Reflector addresses *challenges posed by hallucination and privacy risks.*

consistently excel on simple and medium-difficulty tasks, likely attributable to their strengths in basic analysis and information processing. However, their performance declines on harder tasks, exposing limitations in handling complex IR scenarios. We find that the success of LLMs in IR is not only closely related to the overall difficulty of the task, but also tightly connected to the completion of individual sub-tasks within it, where even subtle differences can precipitate failure. Particularly in complex scenarios, the capabilities of LLMs are frequently constrained by their performance in specific sub-tasks, which ultimately determines the overall IR outcome. For instance, they may falter if it misinterprets a single sub-task requirement, amplifying the impact of such errors in contexts.

To better understand the limitations, we manually investigate and summarize the causes behind the models' sub-task failures, with results detailed in Table 2. Common failure modes include the inability to devise effective IR strategies and to deliver accurate guidance, etc. To elucidate the most critical issues, we provide concrete examples of the two primary causes of failure in Appendix B.1 and B.2. Through manual analysis, we pinpoint the specific reasons driving these failures and outline the key challenges below:

(1) **Difficulty Strategy Formulation**: LLMs frequently struggle to develop effective strategies for IR tasks, particularly in integrating appropriate task prioritization.This limitation stems from their difficulty in synthesizing and organizing the provided information into coherent and actionable plans.
(2) **Inaccurate Guidance or Commands**: Models occasionally generate decisions that are either suboptimal or inaccurate, as well as commands that are erroneous or incompatible with the target operating system, resulting in IR failures.
(3) **Overlooking Minor Details**: During prolonged tasks, LLMs often miss small yet critical details essential for task completion.
(4) **Limited Long-Term Memory**: LLMs face challenges in maintaining the continuity of tasks, managing multiple evidence streams, and preserving coherence over extended contexts.
(5) **Privacy and Security Risks**: In certain instances, LLMs unintentionally process or expose sensitive information during task execution, thereby posing significant potential risks to data privacy and security.

5 IRCOPILOT

5.1 Design Overview

To address the challenges mentioned above, we propose a novel solution named IRCopilot. Our approach is inspired by the operational framework of real-world IR teams, specifically the Blue Team [18]. In these teams, leaders leverage a comprehensive understanding of the infrastructure to formulate strategies, decompose them into manageable sub-tasks, and delegate these to individual executors, orchestrating the entire IR effort. The executors independently carry out their assigned tasks and provide feedback to the leader. The leader evaluates the outcomes, reflects on identified issues, and refines their decisions, thereby iteratively advancing IR.

Based on this workflow, we design IRCopilot with three dynamic phases—Reasoning, Action, and Reflection—and coordinate its operation through four LLM session components: Planner, Generator, Reflector, and Analyst, as illustrated in Figure 2. The Planner formulates the strategic overall plan and decomposes it into manageable sub-tasks; the Generator produces specific instructions or commands based on Planner's directives; the Reflector provides optimization suggestions by reflecting on various types of information; and the Analyst thoroughly analyzes the execution results and feeds them back to Planner. Each phase maintains an independent LLM session to preserve its specific session history and contextual information. Each component performs its designated role without interfering with others, thereby enhancing the efficiency and generalization capabilities of IRCopilot. This structured division of labor, rooted in real-world practices, enables IRCopilot to mitigate LLM limitations and achieve greater precision.

Drawing lessons from failures, we incorporate several innovations into the design of IRCopilot to enhance its practicality and effectiveness. (1) To bolster the reasoning capabilities of LLMs, we employ a range of prompting methodologies (*i.e.*, CoT [51], ToT [52], Few-Shot Prompt [40], and Negative Prompt [45]). (2) We refine

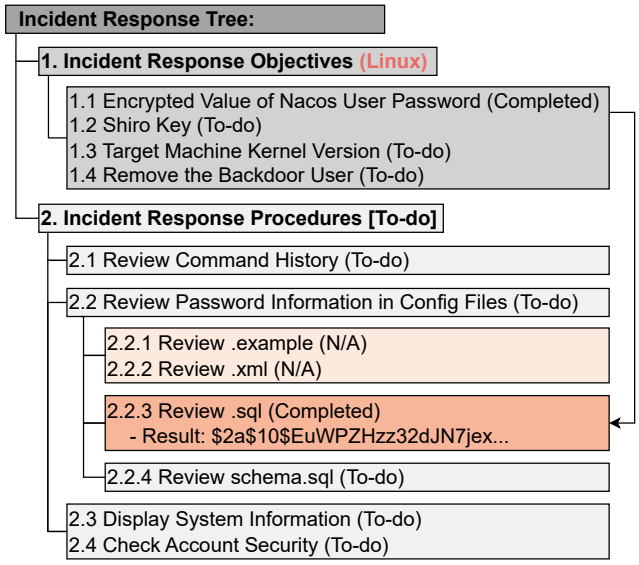


Figure 3: Natural Language Representation of the IRT for the XuanJi-Nacos Target.

the responsibilities of each phase, enabling the LLM to deliver more precise decisions while minimizing the risk of overlooking critical details. (3) To address Challenges I and IV, we introduce an IR Tree (Sec. 5.2), which integrates two key elements: an IR objective table and detailed IR steps. (4) We design the Reflector component (Sec. 5.4) to reflect on four distinct sources of information and offer adaptive optimization suggestions for improved decision-making. The complete prompts are available at our anonymous GitHub [12]. Detailed design specifics for each component are provided below.

5.2 Planner

To overcome the challenges of strategy formulation and the loss of session context, we develop the Planner component. The Planner serves as the core decision-making component in IRCopilot, responsible for overseeing IR tasks at a macro level, making critical decisions, and coordinating the entire IR process. The Planner receives IR outcomes or intents from the user and prepares for subsequent analysis and actions. These steps are then forwarded to the Generator for further detailed interpretation and execution.

IR is typically a continuous and complex process, necessitating the development of long-term strategies and the maintenance of contextual coherence. We draw lessons from Failure Case 1 and inspiration from the Penetration Testing Task Tree [29], motivating the design of IRCopilot and leading to the introduction of the IRT (Incident Response Tree) as a core mechanism for task decomposition and tracking. The IRT tracks task statuses (*i.e.*, "Completed," "To-do," and "N/A") in a structured tree format. This structured approach enhances task clarity at each stage and ensures the continuity of the IR process, thereby mitigating context loss in LLMs.

Taking the XuanJi-Nacos target as an example, we demonstrate the operational mechanism of the Planner and its corresponding prompt strategy (Figure 4), which we divide into reasoning and decision-making processes.

As the leader of cybersecurity incident response, you are responsible for high-level planning and maintaining an Incident Response Tree (IRT). Adhere to the following principles:

1. **Task Structure:**
   - Organize all tasks in a hierarchical sequence (e.g., 1, 1.1, 1.1.1), where sub-tasks are nested under their parent tasks to form a clear tree-like structure.
   - Each level reflects the dependency and granularity of tasks, ensuring logical progression from high-level objectives to specific actions.
2. **Task Status and Updates:**
   - Assign each task a status: "To Do," "Completed," or "N/A," and update it based on the latest findings with a brief outcome report.
   - For tasks under "1. Incident Response Objectives," replace the status with specific details (e.g., answers or information) in parentheses once resolved (e.g., "1.1 OS version - (Ubuntu 20.04)"). Note: "N/A" is not permitted in this section.
   - For other sections (e.g., "2. Incident Response Procedures"), retain the standard status labels and append results separately when applicable.
3. **Adding Sub-Tasks:**
   - Only add sub-tasks if a task is unclear or requires further investigation (e.g., analyzing historical command outputs). Avoid including unverified or undiscovered information in the IRT.
4. **Prohibited Commands:**
   - Avoid global search commands like "find" or "grep."

Below are the IRT templates generated under two different scenarios:

Scenario 1: When tasks are clear (e.g., specific information is required via a defined method), focus solely on "1. Incident Response Objectives" without additional sections:

1. Incident Response Objectives (linux) - [To-do]
1.1 Server OS version - (To-do)
1.2 Sensitive files in home directory - (To-do)
...

Scenario 2: When tasks lack clarity (e.g., involving a "flag field"), expand the IRT with "2. Incident Response Procedures" for further investigation:

1. Incident Response Objectives (linux) - [To-do]
1.1 Attacker IP - (...)
1.2 Modified plaintext admin password - (To-do)
...
2. Incident Response Procedures - [To-do]
2.1 Review Command History - (Completed)
Results from 2.1: - ...
2.2 Investigate Sensitive Directories - (To-do)
...

Figure 4: The prompt strategy of Planner for two scenarios: principles in the initial part, followed by specific strategies for Scenario 1 (clear tasks) and Scenario 2 (unclear tasks).

(1) Initialization and Classification of the IRT ❶: The Planner acquires the user's IR objectives and fundamental system information to construct the initial IRT structure, represented as a hierarchical task table in natural language. We conduct a categorized discussion of user intentions. When clear IR goals are provided, IRCopilot initiates the response using these as the starting point, and logging them in the "*Incident Response Objectives*" table (see Section 1 in Figure 3). For scenarios lacking explicit objectives, we assign priorities to IR sub-tasks based on expert insights from Sec. 4, and maintain a secondary "*Incident Response Procedures*" tree under the root node (see Section 2 in Figure 3). IRCopilot begins the investigation with high-priority tasks, progressively addressing subsequent ones. Throughout this process, the Planner classifies and labels tasks. Simultaneously, it performs a preliminary deduction of potential execution steps within the task table, laying the foundation for subsequent IR processes.

(2) Verification and Maintenance of the IRT ❷: After the IRT is generated, the Reflector verifies it to ensure that irrelevant tasks are excluded, preventing unnecessary objectives from being mistakenly incorporated into the IRT. During the verification process, the Reflector prevents modifications to the root node or key IR objectives of the IRT, thereby avoiding structural deviations or errors in response steps caused by LLM hallucinations. If any issues are detected during verification, the Reflector triggers a rollback mechanism, prompting the Planner to re-evaluate and deduce the IRT to ensure task accuracy and correctness.

(3) Dynamic Sub-task Management ❸: Following the update of the IRT, the Planner prioritizes recently added sub-tasks, incorporating them into the pending task list and formulating concise solutions by weighing various contextual factors. In Figure 3, this is reflected by the prioritization of the recently added 2.2.x sub-tasks. Specifically, the sub-task "2.2.3 Review nacos-mysql.sql" is completed, while tasks such as "2.2.4 Review schema.sql," "2.1 Review Command History," and "2.3 Display System Information" remain in the "To-do" status. ❹ The Planner will evaluate these tasks, determine their priority, and formulate decisions to refine the execution path. Subsequently, the tasks are delegated to the Generator for step-by-step analysis and generation of execution instructions.

(4) Continuous Optimization and Feedback Loop ❺: To optimize the continuous update process of the IRT, we introduce a result recording mechanism at IRT nodes (e.g., Result: $2a$10$...). This ensures that task execution statuses and outputs are updated in real-time within the IRT, combating the forgetting issue associated with LLMs. Additionally, the system conducts real-time reflection on decisions, and the Planner dynamically adjusts IRT branches based on these results, continuously optimizing subsequent task paths to improve IR efficiency.

Furthermore, for ethical and security concerns, we design an interface that enables direct interaction with the Planner to prevent LLM from unintentionally accessing sensitive data.

5.3 Generator

To address the challenges of inaccurate guidance or commands, we designed the Generator. The Generator plays a critical role in guiding and generating the sub-tasks determined by the Planner, responsible for converting abstract task requirements into concrete executable commands. Upon receiving a new sub-task, IRCopilot directs the Generator to disregard extraneous contextual information, focus exclusively on the current sub-task, and avoid interference from other sub-tasks.

In this process, we do not have the Generator produce guidance directly; instead, it decomposes the task into several sequential steps. ❻ Firstly, the Generator focuses on refining the sub-tasks formulated by the Planner into a series of detailed steps. We briefly tag the OS in *IRT-Incident Response Objectives* and prompt the Generator to formulate specific strategies for different OS based on the current context. ❼ When a sub-task can be accomplished in multiple ways, the Generator creates as many distinct guidance strategies as possible. Subsequently, it converts these guidance strategies into executable terminal commands or provides precise investigative descriptions within desktop operating systems. ❽ Throughout this process, the Generator not only ensures that the semantics and logic of the commands align with expectations but also strictly standardizes their format by using the "$" symbol as both the starting and ending markers. This formatting approach facilitates subsequent parsing and extraction of key information while minimizing parsing ambiguities caused by inconsistent formats, thereby enhancing the system's robustness and the accuracy of task execution.

The Generator serves as a pivotal component connecting the forward-looking decisions provided by the Planner with the specific execution steps of IR. It guarantees that strategic plans are carefully converted into accurate and executable operational instructions. This conversion process transforms abstract strategic intentions into concrete practical actions, thereby significantly enhancing the coherence and overall efficiency of IR execution. Furthermore, it produces clear and human-friendly outputs, ensuring the integrity and traceability of the response process.

5.4 Reflector

The Reflector is capable of effectively analyzing and reflecting on the natural language interactions occurring among the other three central components. The rationale of introducing this component is primarily based on the following four needs. First, it mitigates privacy and security concerns, as LLMs may unintentionally access sensitive data, posing significant risks to data security during task execution. Second, contemporary LLMs still suffer from hallucination issues, which may cause other components to understand or generate erroneous information, thereby directly leading to the failure of IR tasks or increasing unnecessary steps and costs. Third, users who lack specialized IR knowledge find it difficult to determine whether the LLM has made suboptimal decisions, generated incorrect guidance, or neglected key information. Finally, in iterative scenarios, the Reflector prevents the accumulation of errors by monitoring and correcting outputs, ensuring other components do not perpetuate initial mistakes when handling new tasks or expanded contexts. Thus, the Reflector is essential for resolving these challenges.

In IRCopilot, the Reflector processes four distinct types of information: (1) ❷ IRT consists of content generated by the Planner and updated by the Analyst; (2) ❺ decisions made by the Planner together with concise descriptions; (3) ❾ guidance or commands

produced by the Generator based on the Planner's decisions and descriptions; and (4) ⓫ execution results of the Generator's guidance. Drawing inspiration from methodologies such as Few-Shot Prompt and Negative Prompt, we develop a set of prompts that empower the Reflector to identify potential hallucinations across contexts, confirm the inclusion of all critical information, pinpoint erroneous steps and their causes, and ultimately suggest modifications and conduct traceback analysis.

5.5 Analyst

IR requires the comprehensive analysis of multiple distinct traces left by attackers, such as intrusion paths, malware activities, indications of data breaches, and anomalous behaviors in system logs. This process takes the form of a one-to-many, potentially weighted, tree structure. Therefore, our IRT design must comprehensively consider the possibilities of various tasks and their inter-dependencies. Additionally, the issue of overlooking minor but crucial details must be taken into account in the design.

To address these issues, we design the Analyst as an auxiliary module to the Planner. It adopts the Tree-of-Thought (ToT) [52] reasoning technique, which generalizes over CoT to prompting language models, and enables exploration over coherent units of thoughts that serve as intermediate steps toward problem solving. ToT enables LLMs to engage in intentional decision-making by evaluating various reasoning trajectories and reflecting on their choices to determine the subsequent actions. Additionally, ToT facilitates forward planning and retrospective adjustments when required, ensuring that decisions are made with a comprehensive and global perspective. The Analyst breaks down the complete results into multiple executable sub-tasks. In particular, once the execution results for specific sub-tasks are received, it dynamically adds items requiring further investigation to the IRT. Finally, the Analyst feeds these findings back to the Planner, helping it make more precise and effective decisions.

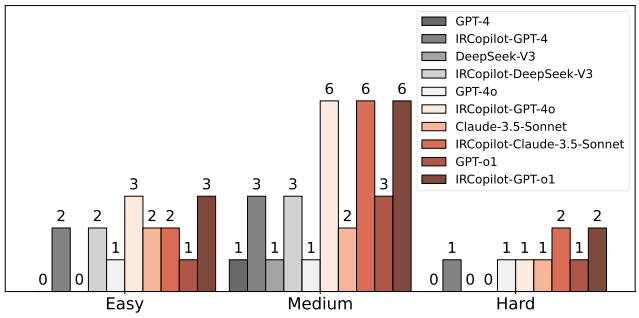
6 EVALUATION ON IRCOPILOT

In this section, we assess the performance of IRCopilot in IR by addressing the following research questions:

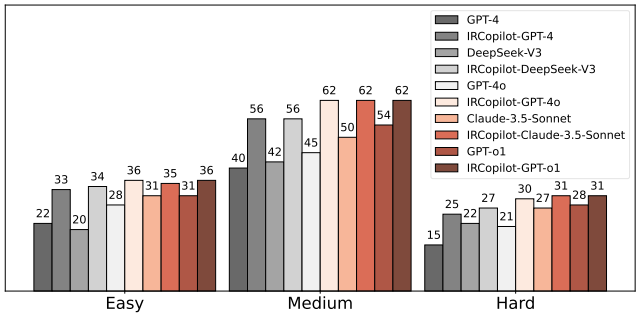
- **RQ1 (Performance & Efficiency):** How does IRCopilot perform with different LLMs in terms of performance and efficiency?
- **RQ2 (Ablation):** What is the contribution of each component within IRCopilot to the overall IR?
- **RQ3 (Practicality):** Does IRCopilot demonstrate practicality and effectiveness in real-world IR tasks?

6.1 Experimental Settings

We implement IRCopilot with 2,788 lines of Python3 code and 257 lines of prompts. We evaluate its performance by utilizing the benchmark tests outlined in Sec. 4 and applying it to real-world IR scenarios. We develop four working versions, IRCopilot-GPT-4, IRCopilot-GPT-4o, IRCopilot-Claude-3.5-Sonnet, and IRCopilot-DeepSeek-V3, using GPT-4 [4], GPT-4o [4], Claude-3.5-Sonnet [3], DeepSeek-V3 [9] as the underlying LLMs respectively. Except for the 257 lines of prompts we designed, all other inputs and experimental settings are identical to those described in Sec. 4.3.



(a) Overall completion status.



(b) Sub-task completion status.

Figure 5: The performance of GPT-4, IRCopilot-GPT-4, DeepSeek-V3, IRCopilot-DeepSeek-V3, GPT-4o, IRCopilot-GPT-4o, Claude-3.5-Sonnet, IRCopilot-Claude-3.5-Sonnet, GPT-o1, and IRCopilot-GPT-o1 on the benchmark.

6.2 Performance & Efficiency Evaluation (RQ1)

Figure 5a presents the performance of GPT-4, IRCopilot-GPT-4, DeepSeek-V3, IRCopilot-DeepSeek-V3, GPT-4o, IRCopilot-GPT-4o, Claude-3.5-Sonnet, IRCopilot-Claude-3.5-Sonnet, GPT-o1 and IRCopilot-GPT-o1 across tasks of varying difficulty. As shown, compared to their base LLMs, IRCopilot shows excellent performance in IR tasks. In particular, IRCopilot-GPT-4o, IRCopilot-Claude-3.5-Sonnet, and IRCopilot-GPT-o1 outperform other models at most difficulty levels, exhibiting a notable advantage in handling IR of moderate or lower complexity. In contrast, IRCopilot-GPT-4 and GPT-4 both trail behind IRCopilot-GPT-4o and GPT-4o, revealing the limitations of the GPT-4-based approach in more complex scenarios. We attribute this discrepancy to the relatively smaller training dataset and narrower incident coverage of GPT-4 compared with GPT-4o, which is a limitation that OpenAI also notes on its website [4]. Additionally, we find that IRCopilot's capabilities scale positively with the performance of baseline LLMs.

Figure 5b shows the performance of the above models on sub-tasks with different degrees of difficulty. As illustrated, IRCopilot-GPT-4, IRCopilot-DeepSeek-V3, IRCopilot-GPT-4o, IRCopilot-Claude-3.5-Sonnet, and IRCopilot-GPT-o1 each outperform their respective base LLMs, achieving 150%, 138%, 136%, 119%, and 114% of the base-model performance, respectively. This outcome confirms that our approach effectively alleviates critical bottlenecks of hallucination, context loss, and difficulties in handling progressively complex questions encountered by LLMs in IR workflows.

Table 3: Efficiency comparison of GPT-4o-2024-08-06 and IRCopilot-GPT-4o-2024-08-06 on the benchmark.

Benchmark	n_g	n_i	$\bar{t}_g$ (s)	$\bar{t}_i$ (s)	$\bar{s}_g$ (s)	$\bar{s}_i$ (s)
Investigating Windows	17	22	**249.9**	278.0	14.7	**12.6**
Linux 1	4	7	**61.6**	193.6	**15.4**	27.7
Web 1	7	7	**69.7**	147.4	**10.0**	21.1
Tardigrade	12	15	**143.7**	201.0	**12.0**	13.4
Ransomware	4	6	**118.0**	159.2	29.5	**26.5**
Web 2	7	11	**54.3**	109.1	**7.8**	9.9
Web 3	7	11	**83.3**	289.3	**11.9**	26.3
Black Pages & Tampering	10	10	**62.0**	112.8	**6.2**	11.3
Windows Miner	5	9	92.0	**68.1**	18.4	**7.6**
Linux 2	7	10	**66.1**	294.0	**9.4**	29.4
Nacos	6	6	**75.9**	119.4	**12.7**	19.9
Where-1S-tHe-Hacker	8	14	**108.2**	157.1	13.5	**11.2**

This further verifies the applicability and effectiveness of our approach. However, these methods still encounter difficulties with high-complexity tasks, primarily due to inherent limitations of LLMs that remain unaddressed. Our approach does not expand the training knowledge base in domains such as attacks, investigations, analysis, or response, which limits improvements in more complex scenarios, as further discussed in Sec. 7.

To assess the efficiency of IRCopilot in real-time IR, we conduct a study on the benchmark to measure response times under controlled conditions. We ensure consistency by performing all experiments at the same time and under identical environmental settings, with other configurations matching those described in previous sections. Likewise, we utilize the GPT-4o-2024-08-06 API to seamlessly integrate IRCopilot-GPT-4o-2024-08-06. For each task in the benchmark, we perform ten experiments to measure the time taken by LLM to generate response content.

Real-world IR involves numerous uncontrollable variables, such as network delays and unpredictable incident scenarios. To mitigate this, we focus on the LLM's reasoning time, defined as the duration to process inputs and produce outputs on platform, as a critical and controllable metric. This approach isolates the LLM's performance, emphasizing its reasoning speed as the key efficiency metric. Our goal is to provide reference indicators for efficiency in real-time IR systems. Table 3 compares the efficiency of GPT-4o and IRCopilot on benchmark. We calculate the average time per sub-task by dividing the total time by the number of completed sub-tasks for a more precise efficiency evaluation. The metrics are defined as follows. (1) n_g and n_i: Number of sub-tasks completed by GPT-4o and IRCopilot. (2) $\bar{t}_g$ and $\bar{t}_i$: Average time to complete each task for GPT-4o and IRCopilot (in seconds). (3) $\bar{s}_g$ and $\bar{s}_i$: Average time per sub-task for GPT-4o and IRCopilot (in seconds), computed as $\bar{t}_g/n_g$ and $\bar{t}_i/n_i$ to reflect efficiency.

Our method generally requires more time across most tasks, although it is faster on 4 out of the 12 benchmark sub-tasks. This stems from the need for multi-step reasoning and the maintenance of an IRT, both of which elevate the time cost. Nevertheless, we consider this trade-off of efficiency for enhanced performance to be justified, as evidenced by a 15% accuracy improvement in 'Ransomware' despite a 20% time increase. We hope that future research can improve efficiency while maintaining performance. Importantly, even with this additional time, our approach remains more efficient than traditional manual IR methods [10, 15].

Table 4: Comparison of IRCopilot-GPT-4o-2024-08-06 and IRCopilot-GPT-o1 on the benchmark.

Benchmark	n_{4o}	n_{o1}	$\bar{t}_{4o}$ (s)	$\bar{t}_{o1}$ (s)	$\bar{c}_{4o}$ (USD)	$\bar{c}_{o1}$ (USD)
Investigating Windows	22	22	**278.0**	914.4	**0.78**	5.90
Linux 1	7	7	**193.6**	463.8	**0.27**	2.06
Web 1	7	7	**147.4**	1,095.5	**0.45**	3.79
Tardigrade	15	15	**201.0**	673.3	**0.43**	3.54
Ransomware	6	6	**159.2**	664.0	**0.23**	2.77
Web 2	11	11	**109.1**	708.7	**0.43**	3.45
Web 3	11	11	**289.3**	1,365.8	**0.65**	6.48
Black Pages & Tampering	10	10	**112.8**	674.5	**0.46**	2.17
Windows Miner	9	9	**68.1**	577.5	**0.35**	1.94
Linux 2	10	**11**	**294.0**	885.7	**0.51**	4.73
Nacos	6	6	**119.4**	630.0	**0.32**	2.81
Where-1S-tHe-Hacker	14	14	**157.1**	690.7	**0.35**	3.88

We compare the key metrics of a reasoning-focused LLM (GPT-o1) and a foundational LLM (IRCopilot-GPT-4o) on the benchmark, as detailed in Table 4. The metrics are defined as follows: (1) n_{4o} and n_{o1}: Number of sub-tasks completed by IRCopilot-GPT-4o-2024-08-06 and IRCopilot-GPT-o1. (2) $\bar{t}_{4o}$ and $\bar{t}_{o1}$: Average time to complete each task for IRCopilot-GPT-4o-2024-08-06 and IRCopilot-GPT-o1 (in seconds). (3) $\bar{c}_{4o}$ and $\bar{c}_{o1}$: Average cost to complete each task for IRCopilot-GPT-4o-2024-08-06 and IRCopilot-GPT-o1 (USD).

The outcomes demonstrate that IRCopilot exhibits versatility across LLMs with different thinking patterns. However, the outcomes also reveal that these patterns lead to varying performance characteristics. In particular, the use of GPT-o1 yields a modest performance improvement, but this enhancement is accompanied by a substantial increase in reasoning time and cost overhead. After a comprehensive evaluation of the trade-offs between performance, time, and cost, we conclude that the combination of IRCopilot and GPT-4o offers the best balance for IR tasks and is the optimal choice at present.

6.3 Ablation Study (RQ2)

To analyze the contribution of each component in our approach to the overall IR process, we conduct an ablation study, design and implement the following four variants:

(1) IRCopilot-no-Planner: the Planner component is disabled, bypassing the IRT strategy and forwarding tasks directly to subsequent components.
(2) IRCopilot-no-Generator: the Generator component is deactivated, allowing other modules to directly produce guidance.
(3) IRCopilot-no-Reflector: the Reflector component is disabled, eliminating the reflection step.
(4) IRCopilot-no-Analyst: the Analyst component and the ToT prompting method are disabled, directing execution results straight to the Planner component.

IRCopilot-GPT-4o and IRCopilot-Claude-3.5-Sonnet demonstrate similar performance in Sec. 6.2. However, IRCopilot-GPT-4o stands out with significantly lower costs, approximately 1/5 of IRCopilot-Claude-3.5-Sonnet's, along with faster response times. Due to its superior cost-effectiveness and efficiency, we consider IRCopilot-GPT-4o the preferred choice. Consequently, all variants

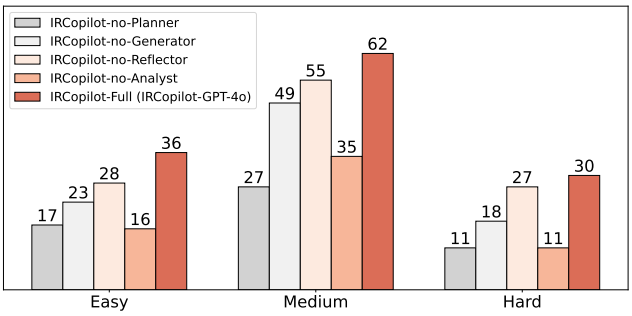


Figure 6: The performance of IRCopilot, IRCopilot-no-Planner, IRCopilot-no-Generator, IRCopilot-no-Reflector and IRCopilot-no-Analyst on the benchmark.

utilize the IRCopilot-GPT-4o API for evaluation. Notably, a comprehensive IR task encompasses multiple objectives, and success requires fulfilling all of them. Conducting ablation experiments on the components could therefore hinder successful task completion. For this reason, our ablation study focuses solely on the quantitative analysis of sub-tasks.

Figure 6 illustrates the performance of the four variants assessed through our comprehensive benchmark tests. Across these evaluations, IRCopilot consistently surpasses each ablation baseline in sub-task completion rates. The ablation study yields several key observations, summarized as follows:

- IRCopilot-no-Planner achieves the lowest success rate, completing only 42.97% (55/128) of the sub-tasks compared to the full variant, followed by IRCopilot-no-Analyst at 48.44% (62/128). These figures fall below even the baseline GPT-4o's performance. This highlights the critical importance of the Planner component's IRT strategy and decision-making capabilities, as well as the Analyst component's ToT method, in ensuring effective IR processes. Moreover, these results emphasize that robust reasoning ability is indispensable for driving successful outcomes in such tasks.
- IRCopilot-no-Generator performs approximately on par with the baseline GPT-4o at 70.31% (90/128), indicating that this component primarily generates IR guidance through the base LLM. This provides auxiliary support to incident responders and reduces operational costs.
- IRCopilot-no-Reflector demonstrates a slight decrease compared to the full variant in sub-task completion at 85.94% (90/128). This decline is mainly attributed to occasional LLM 'hallucinations' or other minor issues. However, the frequency of such errors is relatively low, and the Reflector component is capable of rectifying these minor errors in most cases, thereby positively contributing to the overall IR process. Nevertheless, IRCopilot-no-Reflector still marginally outperforms the baseline GPT-4o, suggesting that even in the absence of the Reflector component, the other components of IRCopilot continue to provide a certain performance advantage.

6.4 Practicality Study (RQ3)

To evaluate the adaptability of IRCopilot in addressing real-world cybersecurity incidents, we conducted a practicality study that extends beyond the benchmark tests outlined in Sec. 4. For this assessment, we selected 35 distinct machines across three different platforms to test IRCopilot's performance:

(1) TryHackMe [19]: A subscription-based cybersecurity education platform. We selected 17 machines of varying difficulty: 7 easy, 9 medium, and 1 hard target.
(2) XuanJi [21]: A pay-per-time platform featuring globally recognized IR machines. From this, we select 12 representative machines, consisting of 2 easy, 6 medium, and 4 hard targets.
(3) ZGSF [22]: A public account platform managed by a cybersecurity laboratory, providing free access. From this, we select 6 representative machines, including 2 easy, 3 medium, and 1 hard target. These machines are adapted from real-world cyber attack cases, with relevant privacy information removed to prevent ethical and moral issues.

Our evaluation integrates IRCopilot with the GPT-4o-2024-08-06 API to create IRCopilot-GPT-4o-2024-08-06, establishing the completion of the entire IR process as the criterion for a successful test. For each target, we conduct five experiments and record the number of times the tasks are successfully completed. *We establish that if at least one of the five experiments results in a successful response, we consider the IR successful.* This is because in real-world cybersecurity IR process, multiple security practitioners usually work individually on one particular workload to perform extended analyses and responses. Therefore, the ultimate success contingent upon the successful completion of the task from at least one of the security experts.

Table 5: IRCopilot performance over TryHackMe challenges.

Challenge	Difficulty	OS	Completions	Cost (USD)
Eradication & Remediation	easy	Linux	4/5 (✓)	0.29
Investigating Windows	easy	Windows	4/5 (✓)	0.78
Linux Incident Surface	easy	Linux	3/5 (✓)	0.53
Linux File System Analysis	easy	Linux	5/5 (✓)	0.46
Linux Process Analysis	easy	Linux	5/5 (✓)	0.69
Threat Intel & Containment	easy	Linux	5/5 (✓)	0.08
Windows Network Analysis	easy	Windows	5/5 (✓)	0.15
Blizzard	medium	Windows	0/5 (✗)	-
Investigating Windows 2.0	medium	Windows	4/5 (✓)	1.11
Investigating Windows 3.x	medium	Windows	4/5 (✓)	0.80
Linux Logs Investigations	medium	Linux	5/5 (✓)	0.29
Linux Live Analysis	medium	Linux	4/5 (✓)	0.54
Tardigrade	medium	Linux	3/5 (✓)	0.43
Tempest	medium	Windows	0/5 (✗)	-
Windows Applications Forensics	medium	Windows	4/5 (✓)	0.31
Windows Event Logs	medium	Windows	5/5 (✓)	0.24
Squid Game	hard	Linux	0/5 (✗)	-

Table 6: IRCopilot performance over XuanJi challenges.

Challenge	Difficulty	OS	Completions	Cost (USD)
Webshell Detection and Removal	easy	Linux	4/5 (✓)	0.24
Windows EVTX File Analysis	easy	Windows	5/5 (✓)	0.21
Linux Backdoor Emergency	medium	Linux	2/5 (✓)	0.80
Log Analysis - MySQL	medium	Linux	0/5 (✗)	-
Log Analysis - Redis	medium	Linux	0/5 (✗)	-
Traffic Feature Analysis - Tomcat	medium	Linux	3/5 (✓)	0.24
VulnTarget n - Ransomware	medium	Linux	4/5 (✓)	0.23
Windows Black Pages & Tampering	medium	Windows	5/5 (✓)	0.46
Memory Trojan Analysis - Nacos	hard	Linux	5/5 (✓)	0.32
Memory Trojan Analysis - shiro	hard	Linux	5/5 (✓)	0.19
Windows Wordpress	hard	Windows	3/5 (✓)	0.26
Where 1S tHe Hacker	hard	Windows	0/5 (✗)	-

Table 7: IRCopilot performance over ZGSF challenges.

Challenge	Difficulty	OS	Completions	Cost (USD)
Linux 1	easy	Linux	5/5 (✓)	0.27
Windows 1	easy	Windows	5/5 (✓)	0.45
Windows 2	medium	Windows	3/5 (✓)	0.43
Windows 3	medium	Windows	4/5 (✓)	0.65
Cryptojacking	medium	Windows	5/5 (✓)	0.35
Linux 2	hard	Linux	0/5 (✗)	-

Table 8: Performance over real-world attack cases.

Machine	OS	Tasks	Completions	Cost (USD)
Sunlogin	Windows	3	4/5 (✓)	0.66
algo	Linux	4	3/5 (✓)	0.51
kswapd0	Linux	3	5/5 (✓)	0.25
Cryptojacking 1	Linux	5	2/5 (✓)	0.90
Cryptojacking 2	Linux	6	4/5 (✓)	0.61

Table 5 details the performance of IRCopilot in the TryHackMe challenges. IRCopilot successfully completes all 7 easy tasks and 7 of 9 medium tasks, achieving a completion rate of 77.8%. Concurrently, when utilizing the GPT-4o-2024-08-06 API, IRCopilot maintains a low operational cost. This indicates that IRCopilot is capable of efficiently handling a wide range of low to medium difficulty of IR tasks with reduced expenditure. Notably, the platform offers few high-difficulty tasks, and only one is available, which IRCopilot does not complete.

Table 6 outlines the performance of IRCopilot on the XuanJi platform challenges. At the easy level, IRCopilot completes all tasks, demonstrating high reliability. For medium challenges, it resolves 3 of 6 tasks (50% completion rate) at an average cost of $0.39 per task. At the hard level, it completes 2 of 4 challenges (50% completion rate), showcasing its capability for complex tasks. In general, IRCopilot exhibits strong adaptability, although performance in higher-difficulty tasks could be improved.

Table 7 presents IRCopilot's performance on ZGSF challenges. It achieves a 100% completion rate for both easy and medium tasks, with average costs of $0.36 and $0.48 per task, respectively. However, IRCopilot fails to complete the hard-level tasks, indicating certain inherent limitations in handling highly complex IR scenarios.

Notably, we included fewer tasks of Hard difficulty. This is due to the limited number of publicly available IR tasks on the platform. Despite this constraint, we ensured that the selected tasks were sufficient to support our conclusions. Specifically, compared to the other two difficulty levels, IRCopilot exhibits slightly lower performance on Hard difficulty tasks.

Additionally, we select five real-world instances of personal PCs or servers compromised by attacks, which are published on the XuanJi platform. These cases span diverse system types, attack scenarios, and techniques. Similarly, to adhere to ethical guidelines, the privacy data of these machines has been anonymized.

Table 8 showcases capabilities of IRCopilot in adapting to tasks on compromised machines within real-world attack environments, demonstrating its effectiveness in identifying and extracting attacker traces and information from actual cyber incidents. This capability is crucial for improving the efficiency of handling and responding to cybersecurity incidents. The findings show that IRCopilot performs robustly and shows considerable potential across a variety of real and diverse challenges, providing valuable support for post-IR and bolstering defenses against the evolving cyber threat landscape.

Table 9: Comparison of IR benchmark release dates against LLM knowledge cutoff dates.

Model	GPT-4	DeepSeek-V3	GPT-4o	Claude-3.5-Sonnet	GPT-o1
Cutoff	2023-04	2024-07	2023-10	2024-04	2023-10
After	10	0	10	3	10
Before	2	12	2	9	2

7 DISCUSSION

Data Contamination Mitigation. IRCopilot is built based on LLMs, recognizing that cases or writeups on these platforms may already be included in LLM training data, which can potentially bias experimental results. To address concerns about data contamination in LLMs/IRCopilot, we implement three measures. Initially, we prioritize benchmark cases with more recent publication dates during the selection process, which are most likely to fall outside the scope of the LLMs' training data. we compare the publication date of each selected case with the specific knowledge cutoff dates of the LLMs [14, 16] as shown in Table 9. Majorities (33/60) of the tasks are after the cutoff date of the models. Subsequently, we assess whether the LLMs underwent targeted training by querying them for detailed information about specific cases. More details are available at our Anonymous Github [12]. Finally, since IRCopilot has the ability to display each step of its reasoning process, we confirm through an examination of its reasoning paths that it lacks prior knowledge of the cases. Furthermore, real-world cases in practicality study further demonstrated that, even without targeted training, IRCopilot shows significant feasibility and effectiveness in real-world applications.

Selection of LLM API. During the design phase, we systematically evaluated multiple LLM APIs and ultimately select four: GPT-4, GPT-4o-2024-08-06, and Claude-3.5-Sonnet as commercial APIs, alongside DeepSeek-V3 as the open-source model. In our experiments, the latest model GPT-4o-2024-08-06 shows even better performance than GPT-4, offering faster response times and a context window of up to 128K tokens. Meanwhile, Claude-3.5-Sonnet and DeepSeek-V3 excel at reasoning and code analysis. Notably, GPT-4o-2024-08-06 also costs just one-twelfth of GPT-4, making it a cost-effective choice for real-world applications.

Limitations. Even advanced models display limitations under specific conditions. For instance, they may produce outputs that contradict prior context, resulting in hallucinations [55, 57]. In our system, these hallucinations typically appear as erroneous analyses, judgments, or decisions derived from existing information (Appendix B.3). To address this, we introduce the IRT and Reflector to mitigate their impact. Beyond hallucinations, complex sample analysis and reverse engineering tasks often remain incomplete due to the limited capabilities of both IRCopilot and LLMs in handling intricate scenarios (Appendix B.4). As training data and model architectures continue to evolve, we anticipate that LLM performance

will improve, gradually reducing the occurrence of hallucinations and enhancing the ability to manage complex IR tasks.

Future Work for Improvement. Despite our efforts to design and evaluate multiple countermeasures, cybersecurity is always an evolving game, as we face the emergence of increasingly diverse and sophisticated attack vectors. We attempted to integrate RAG into the design, but it led to decreased efficiency and higher overhead, failing to yield the anticipated performance gains. Looking forward, a promising research direction is the development of adaptive defense mechanisms. These should incorporate more advanced agent systems, augmented with enhanced memory or search capabilities, to effectively counter sophisticated threats. Our findings indicate a positive correlation between the effectiveness of this approach and the capabilities of baseline LLMs. We are confident that, as LLMs evolve and defense strategies improve, these systems will become more adaptable, robust, and efficient, capable of addressing a broader spectrum of complex security incidents.

8 CONCLUSION

Modern cybersecurity landscape brings numerous challenges for automated incident response. This paper performs a systematic study to address them with two contributions. First, we establish a new comprehensive incident response benchmark to demonstrate the limitations of directly applying LLMs in this complex domain. Second, we design IRCopilot, an advanced LLM-driven approach for efficient incident response by simulating the planning and response processes of real-world professional teams. This is achieved by an interactive modular design, and adoptions of multiple strategies to mitigate LLMs' hallucinations and context limitations. We evaluate IRCopilot through extensive experiments and real-world scenarios, demonstrating the feasibility of leveraging LLMs for incident response tasks and laying the groundwork for future developments of cybersecurity enhancement.

REFERENCES

[1] 2025. *The 5/5/5 Benchmark.* Retrieved March 24, 2025 from https://sysdig.com/555-benchmark
[2] 2025. *7 Incident Response Metrics and How to Use Them.* Retrieved March 24, 2025 from https://securityscorecard.com/blog/how-to-use-incident-response-metrics
[3] 2025. *Anthropic.* Retrieved March 24, 2025 from https://docs.anthropic.com
[4] 2025. *API platform.* Retrieved March 24, 2025 from https://openai.com/api
[5] 2025. *Certified Information Systems Security Professional.* Retrieved March 24, 2025 from https://www.isc2.org/certifications/cissp
[6] 2025. *Check Point Software.* Retrieved March 24, 2025 from https://www.checkpoint.com
[7] 2025. *Claude.* Retrieved March 24, 2025 from https://claude.ai
[8] 2025. *DeepSeek.* Retrieved March 24, 2025 from https://www.deepseek.com
[9] 2025. *DeepSeek API Docs.* Retrieved March 24, 2025 from https://api-docs.deepseek.com
[10] 2025. *Grey Time: The Hidden Cost of Incident Response.* Retrieved April 13, 2025 from https://www.rapid7.com/blog/post/2022/09/13/grey-time-the-hidden-cost-of-incident-response/
[11] 2025. *Human-in-the-loop.* Retrieved March 24, 2025 from https://en.wikipedia.org/wiki/Human-in-the-loop
[12] 2025. *IRCopilot: Automated Incident Response with Large Language Models.* Retrieved March 24, 2025 from https://anonymous.4open.science/r/IRCopilot-78A0
[13] 2025. *kaspersky.* Retrieved March 24, 2025 from https://www.kaspersky.com
[14] 2025. *Knowledge Cutoff Dates of all LLMs explained.* Retrieved March 24, 2025 from https://otterly.ai/blog/knowledge-cutoff
[15] 2025. *Mean Time to Respond: Optimizing IT Performance.* Retrieved April 13, 2025 from https://blog.invgate.com/mean-time-to-respond
[16] 2025. *Models - OpenAI API.* Retrieved March 24, 2025 from https://platform.openai.com/docs/models
[17] 2025. *National Vulnerability Database.* Retrieved March 24, 2025 from https://nvd.nist.gov
[18] 2025. *Red Team vs Blue Team in Cybersecurity.* Retrieved March 24, 2025 from https://www.offsec.com/blog/red-team-vs-blue-team
[19] 2025. *TryHackMe.* Retrieved March 24, 2025 from https://tryhackme.com
[20] 2025. *Unit 42.* Retrieved March 24, 2025 from https://unit42.paloaltonetworks.com
[21] 2025. *XuanJi.* Retrieved March 24, 2025 from https://xj.edisec.net
[22] 2025. *ZGSF Lab.* Retrieved March 24, 2025 from https://mp.weixin.qq.com/mp/profile_ext?action=home&__biz=MzkxMTUwOTY1MA==
[23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
[24] Md Tanvirul Alam, Dipkamal Bhusal, Youngja Park, and Nidhi Rastogi. 2023. Looking beyond IoCs: Automatically extracting attack patterns from external CTI. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses.* 92–108.
[25] Maryam Amirizaniani, Jihan Yao, Adrian Lavergne, Elizabeth Snell Okada, Aman Chadha, Tanya Roosta, and Chirag Shah. 2024. Developing a framework for auditing large language models using human-in-the-loop. *arXiv preprint arXiv:2402.09346* (2024).
[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.
[27] Yiren Chen, Mengjiao Cui, Ding Wang, Yiyang Cao, Peian Yang, Bo Jiang, Zhigang Lu, and Baoxu Liu. 2024. A survey of large language models for cyber threat detection. *Computers & Security* (2024), 104016.
[28] KR1442 Chowdhary and KR Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.
[29] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. 2024. {PentestGPT}: Evaluating and harnessing large language models for automated penetration testing. In *33rd USENIX Security Symposium (USENIX Security 24).* 847–864.
[30] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[31] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
[32] Mohamed Amine Ferrag, Mthandazo Ndhlovu, Norbert Tihanyi, Lucas C Cordeiro, Merouane Debbah, and Thierry Lestable. 2023. Revolutionizing cyber threat detection with large language models. *arXiv preprint arXiv:2306.14263* (2023).
[33] Joint Task Force. 2017. *Security and privacy controls for information systems and organizations.* Technical Report. National Institute of Standards and Technology.
[34] Junjie Huang and Quanyan Zhu. 2024. PenHeal: A Two-Stage LLM Framework for Automated Pentesting and Optimal Remediation. In *1st International Workshop on Autonomous Cybersecurity, AutonomousCyber 2024, As part of the 31st ACM Conference on Computer and Communications Security, ACM CCS 2024.* Association for Computing Machinery, Inc, 11–22.
[35] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
[36] Jia Li, Ge Li, Chongyang Tao, Huangzhao Zhang, Fang Liu, and Zhi Jin. 2023. Large language model-aware in-context learning for code generation. *arXiv preprint arXiv:2310.09748* (2023).
[37] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. Pre-trained language models for text generation: A survey. *Comput. Surveys* 56, 9 (2024), 1–39.
[38] Lu Li and Bojie Gong. 2023. Prompting Large Language Models for Malicious Webpage Detection. In *2023 IEEE 4th International Conference on Pattern Recognition and Machine Learning (PRML).* IEEE, 393–400.
[39] Peiyu Liu, Junming Liu, Lirong Fu, Kangjie Lu, Yifan Xia, Xuhong Zhang, Wenzhi Chen, Haiqin Weng, Shouling Ji, and Wenhai Wang. 2024. Exploring {ChatGPT's} capabilities on vulnerability management. In *33rd USENIX Security Symposium (USENIX Security 24).* 811–828.
[40] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837* (2022).
[41] Alexander Nelson, Sanjay Rekhi, Murugiah Souppaya, and Karen Scarfone. 2024. *Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile.* Technical Report. National Institute of Standards and Technology.
[42] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2022. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP).* IEEE, 754–768.

[43] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. 2023. Examining zero-shot vulnerability repair with large language models. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2339–2356.
[44] Jiaxing Qi, Shaohan Huang, Zhongzhi Luan, Shu Yang, Carol Fung, Hailong Yang, Depei Qian, Jing Shang, Zhiwen Xiao, and Zhihui Wu. 2023. Loggpt: Exploring chatgpt for log-based anomaly detection. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 273–280.
[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
[46] Samaneh Shafee, Alysson Bessani, and Pedro M Ferreira. 2024. Evaluation of llm chatbots for osint-based cyber threat awareness. *arXiv preprint arXiv:2401.15127* (2024).
[47] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2023), 8634–8652.
[48] Chengyu Song, Linru Ma, Jianming Zheng, Jinzhi Liao, Hongyu Kuang, and Lin Yang. 2024. Audit-LLM: Multi-Agent Collaboration for Log-based Insider Threat Detection. *arXiv preprint arXiv:2408.08902* (2024).
[49] Wannita Takerngsaksiri, Jirat Pasuksmit, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Ruixiong Zhang, Fan Jiang, Jing Li, Evan Cook, Kun Chen, and Ming Wu. 2024. Human-In-the-Loop Software Development Agents. *arXiv preprint arXiv:2411.12924* (2024).
[50] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019. Learning Deep Transformer Models for Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1810–1822.
[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.
[52] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems* 36 (2023), 11809–11822.
[53] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.
[54] Shuhan Yuan and Xintao Wu. 2021. Deep learning for insider threat detection: Review, challenges and opportunities. *Computers & Security* 104 (2021), 102221.
[55] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. 2023. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534* (2023).
[56] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics* 12 (2024), 39–57.
[57] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the AI ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219* (2023).

A GLOBAL CYBERSECURITY LANDSCAPE

B CASE DISCUSSION

B.1 Case I (False IR Strategy)

In Task 2 of the benchmark "ZGSF_Linux 1", the "False Response Strategy" results in a typical failure.

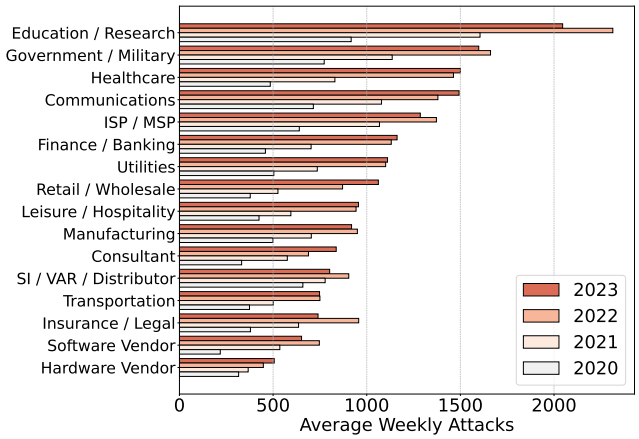
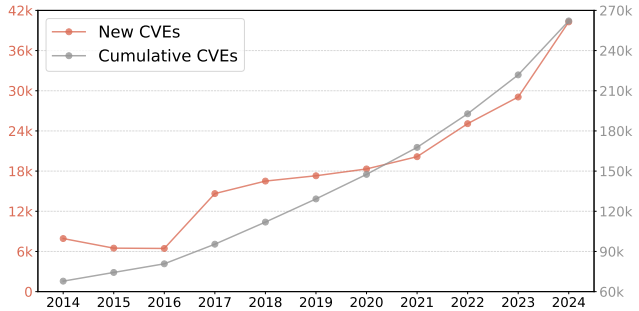
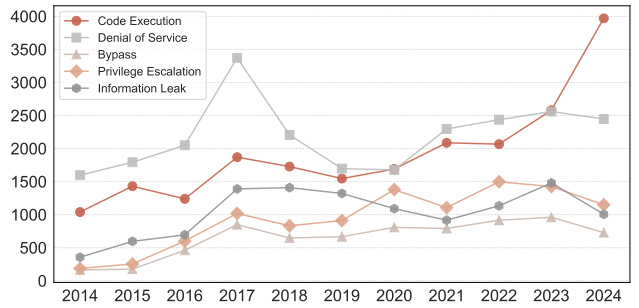


Figure 7: Average Weekly Attacks per Organization by Industry (2020-2023).



(a) CVE Trends (2014-2024).



(b) Vulnerabilities by Impact Types (2014-2024).

Figure 8: Overview of CVEs (2014-2024).

False IR Strategy (LLMs)

Use the following methods to systematically identify potential flag information in the system:
1. Search for all possible flag files in the system:
`find / -type f -name "flag*" 2>/dev/null`
2. Inspect memory or processes for hidden information:
`strings /dev/mem | grep -i "flag"`
3. Analyze network traffic and scheduled tasks:
`tcpdump -A -i eth0 | grep "flag"`
4. Examine attack scripts and environment variables:
`env | grep "flag"`

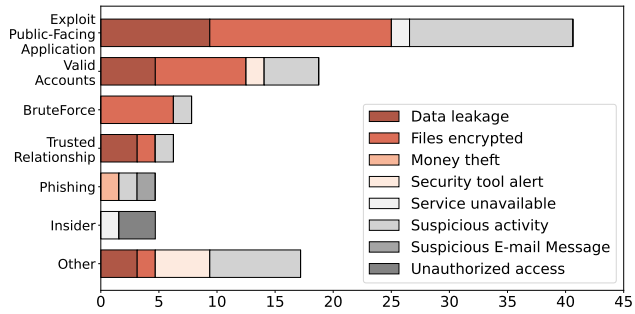


Figure 9: Security Vulnerabilities Introduced by Various Attack Vectors

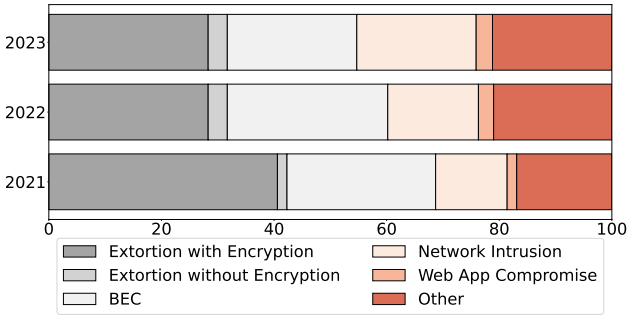


Figure 10: Top 5 Investigation Types in Unit 42 IR Cases (2021-2023).

IR should adopt a **step-by-step clue analysis process** instead of relying solely on keyword searches, as some clues may lie beyond the files. This highlights a key limitation of LLMs in devising effective strategies for IR tasks. To address this, we introduced IRT in IRCopilot to prioritize IR tasks. When the task lacks clear information, it will be executed sequentially in accordance with the order in the "Incident Response Procedures."

IRCopilot-Planner

IRT:
1. Incident Response Objectives (Linux) - [To-do]
   1.1 Hacker's IP address - (To-do)
   1.2 Flag 1 - (To-do)
   ...
2. Incident Response Procedures - [To-do]
   2.1 Review Command History - (To-do)
   2.2 Investigate Sensitive Directories - (To-do)
   2.3 Analyze System Logs - (To-do)
   ...

Task selection: **2.1 Review Command History**
1. View the command history of the current user.
   `$ history $`
2. Looking directly at the ".bash_history" file may provide more information, especially after the user has logged out.
   `$ cat ~/.bash_history $`

B.2 Case II (False Commands Generation)

We present a specific example of a failure caused by the "False Command Generation" in Task 6 of "XuanJI_Where-1S-tHe-Hacker" within the benchmark. When searching for hidden accounts, directly using the "net user" command fails to provide complete information. This case shows that LLMs may generate incorrect commands, which contradict facts and lead to task failures.

False Commands Generation (LLMs)

Hidden Accounts Created by Hackers
Check the system account list:
1. Run `net user` to view the list of accounts and check for any suspicious accounts.
2. Run `net user <username>` to view detailed information.

In IRCopilot, the approach should involve first analyzing subtasks and then generating multiple solutions, specifically by **checking Regedit or using the WMIC command**.

IRCopilot-Generator

To complete the task, we need to check in detail. Here are the specific steps and commands to perform this task:
1. Hidden accounts are typically recorded in the registry at the following location:
   `$ ...\SAM\SAM\Domains\Account\Users $`
This path stores all user SIDs and related information.
2. The WMIC can quickly list all accounts and their SIDs:
   `$ wmic useraccount get name,sid $`

B.3 Case III (Hallucinations)

We present an example of "XuanJI_Where-1S-tHe-Hacker" in the benchmark, specifically Task 9, which involves identifying the moment the hacker reads the stored key from the file. This task remains unsolved by both LLMs and IRCopilot due to hallucinations. The suggested Event IDs 4656 and 4663 track general file access events, which are too broad and may not specifically capture cryptographic key operations. This can lead to missed or irrelevant logs, failing to pinpoint the exact moment of the key access. In contrast, **Event ID 5058 is the appropriate choice**, as it logs key file operations (*e.g.*reading a stored cryptographic key) in the Windows Security event log, enabling accurate identification of the event.

False Guidance Generation (Hallucinations)

"The moment the hacker reads the stored key from the file" requires checking the following:
1. Check the file access events related to the key (event ID 4656 or 4663)
2. Check the file access records under sensitive directories

B.4 Case IV (Limitations)

Here, we offer an example task highlighting the limitations of LLMs and IRCopilot: the Squid Game malicious file analysis challenge from TryHackMe. This challenge demands sophisticated dynamic and static analysis, encompassing data flow analysis, VBA macro handling, obfuscated content decoding, and the seamless integration of multiple tools.

Results of Oledump

```
ubuntu@ip-10-10-114-110:~/Desktop/maldocs$ oledump.py attacker1.doc
1:         114 '\x01CompObj'
...
8: M    9852 'Macros/VBA/ThisDocument'
...
```

LLMs

```
oledump.py -s 8 -v attacker1.doc
oledump.py -s 1 -d attacker1.doc
```

The output from these two commands generates obfuscated data streams that are challenging for LLMs to parse, often necessitating intricate manual analysis. We're optimistic that future developments will address this.

C ETHICAL CONSIDERATIONS AND OPEN SCIENCE POLICY COMPLIANCE

C.1 Ethical Considerations

This research investigates automated incident response with LLMs. While we believe our work can contribute to incident response in real-world environments, we adhere to key ethical principles to minimize potential risks:

- **Risk Considerations:** Effective corrective measures require that the model remains under the victim's control. To achieve this, we deploy the incident response model on a separate, independent machine, ensuring continuous victim oversight and minimizing the risk of malicious manipulation. This strategy safeguards the security of LLMs and is essential for their safe and reliable deployment.
- **Data Privacy Protection:** To prevent the model from directly accessing sensitive personal data, we employ professionals from the victim's side as intermediaries and have the model operate on an isolated machine. This approach safeguards the privacy of all parties and ensures data security. Additionally, we equip IRCopilot with interfaces to access local LLMs and intentionally design the model to avoid accessing system private data and services unless absolutely necessary.
- **Promotion of Ethical Guidelines:** We firmly oppose any malicious use of IRCopilot and actively advocate for ethical guidelines to ensure it consistently serves a positive role in the cybersecurity landscape.

Through this work, we aim to make a positive contribution to the practical application of LLMs in cybersecurity incident response, assisting IT resource owners in mitigating the impact of cyberattacks.

C.2 Compliance with the Open Science Policy

We are committed to the principles of open science, ensuring that our research outputs are publicly accessible. This includes:

- **Open Source Code.** We fully endorse open science principles and are dedicated to promoting transparency, reproducibility, and collaboration in scientific research. The full source code for our project will be openly available on Anonymous GitHub at https://anonymous.4open.science/r/IRCopilot-78A0 after publication. This allows for transparency, reproducibility, and enables other researchers to build upon our work.
- **Benchmark Access.** We publish our benchmark in an anonymized repository https://anonymous.4open.science/r/IRCopilot-78A0. This provides a fair and comprehensive evaluation for incident response and supports the phased comparison of research outcomes.

By sharing both our code and benchmark, we seek to promote academic collaboration, technological advancement, and reproducibility in the field.

Table 10: Summarized 27 types of sub-tasks in the proposed incident response benchmark.

Phase	Technique	Description
Detection	System Information Gathering	Includes operating system identification, network configuration analysis, hardware information gathering, etc.
	Open Port Identification	Detect open network ports on the target system.
	Service Enumeration	Identify and analyze running services to uncover version details and vulnerabilities.
	Directory Inspection	Examine key directories and hidden files for unusual activity.
	Account Security Review	Audit user account permissions and identify unauthorized accounts or backdoors.
	File Integrity Check	Monitor file hashes to detect unauthorized changes.
	Other Detections	Inspect other vulnerable areas of the system, such as the registry, etc.
Response	Historical Command and Behavior Analysis	Review user commands and system behaviors to detect abnormal operations.
	Permission Review and Management	Audit system and application permissions to enforce least-privilege principles and manage risky permissions.
	File Analysis	Inspect system files and source code for vulnerabilities or malicious content.
	Malicious File Handling	Identify, isolate, and remove malicious files to prevent further damage.
	Startup Item Analysis	Review startup items for unauthorized programs or scripts.
	Scheduled Task Analysis	Analyze the system's scheduled tasks settings to identify possible malicious or planned tasks.
	Anomaly Behavior Response	Respond to abnormal system or user behaviors to contain potential threats.
	Memory and Process Analysis	Examine memory and processes to identify abnormal or malicious activity.
	Malicious Process Handling	Terminate malicious processes to mitigate ongoing threats.
	System Log Analysis	Analyze system logs for signs of compromise, unauthorized access, or suspicious activities.
	Application Log Analysis	Review application logs for exploitation attempts or unusual behavior.
	Network Traffic Analysis	Analyze network traffic to identify suspicious communication or data exfiltration.
	Risky IP Management	Block or monitor traffic from known malicious or suspicious IP addresses.
	Database Analysis	Analyze databases for security vulnerabilities, data leaks, or unauthorized access.
	Other Responses	Conduct additional response activities, such as analyzing virtualization environments or reviewing container security.
Recovery	System Recovery	Restore the system to a stable state after failures, malware, or misconfigurations.
	Data Recovery	Recover lost or corrupted data from backups or damaged media.
	Service Recovery	Restore key services and applications to minimize downtime.
	Vulnerability Patching	Apply patches to fix vulnerabilities and prevent recurrence of attacks.
	Other Recoveries	Additional recovery methods, such as network recovery and permission reset, to address various aspects of system restoration.