Benchmarking LLMs in an Embodied Environment for Blue Team Threat Hunting

Xiaoqun Liu[1], Feiyang Yu[2], Xi Li[3], Guanhua Yan[4], Ping Yang[4], and Zhaohan Xi[4]

[1]Southern University of Science and Technology
[2]Duke University
[3]University of Alabama at Birmingham
[4]Binghamton University
[1]liuxq2021@mail.sustech.edu.cn, [2]fy66@duke.edu, [3]XiLiUAB@uab.edu, , [4]{ghyan, pyang, zxi1}@binghamton.edu

Abstract

As cyber threats continue to grow in scale and sophistication, blue team defenders increasingly require advanced tools to proactively detect and mitigate risks. Large Language Models (LLMs) offer promising capabilities for enhancing threat analysis. However, their effectiveness in real-world blue team threat-hunting scenarios remains insufficiently explored. In this paper, we present CYBERTEAM, a benchmark designed to guide LLMs in blue teaming practice. CYBERTEAM constructs an embodied environment in two stages. First, it models realistic threat-hunting workflows by capturing the dependencies among analytical tasks from threat attribution to incident response. Next, each task is addressed through a set of embodied functions tailored to its specific analytical requirements. This transforms the overall threat-hunting process into a structured sequence of function-driven operations, where each node represents a discrete function and edges define the execution order. Guided by this framework, LLMs are directed to perform threat-hunting tasks through modular steps. Overall, CYBERTEAM integrates 30 tasks and 9 embodied functions, guiding LLMs through pipelined threat analysis. We evaluate leading LLMs and state-of-the-art cybersecurity agents, comparing CYBERTEAM's embodied function-calling against fundamental elicitation strategies. Our results offer valuable insights into the current capabilities and limitations of LLMs in threat hunting, laying the foundation for the practical adoption in real-world cybersecurity applications.

1 Introduction

The increasing frequency and sophistication of cyber threats continue to pose significant challenges to organizational security. In 2024 alone, over $11,000$ more (38% increase!) vulnerabilities were reported compared to 2023, as evidenced by the MITRE CVE database [87]. Defenders, commonly known as the **blue team** [25, 71], are under increasing pressure to identify, analyze, and respond to malicious activities in a timely and accurate manner, a process termed as **threat hunting**. Traditionally, threat hunting has been a labor-intensive process, relying heavily on the expertise of analysts to sift through logs, correlate indicators of compromise (IOCs), and construct hypotheses about potential attacks [21, 33]. This process demands both deep domain knowledge and an ability to integrate fragmented evidence from multiple sources under time constraints [8, 97, 102].

Recent advances in Large Language Models (LLMs) have demonstrated impressive potential to augment cybersecurity practices, including malware analysis [2, 5, 70, 24], penetration testing [22, 23, 36, 63], and fuzzing [99, 66, 10]. Building on this progress, there is growing interest in

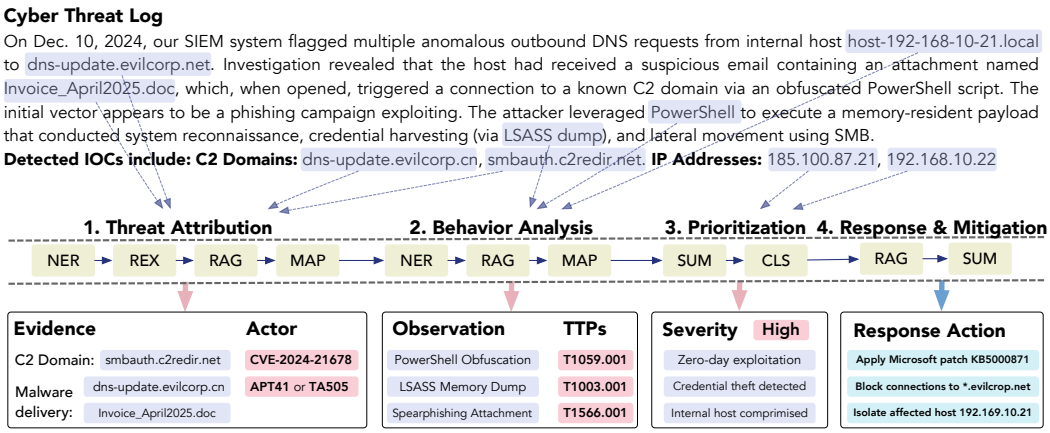

Figure 1: A threat hunting example equipped with the embodied functions. Function names: NER–named entity recognition, REX–regex parsing, MAP–text mapping, RAG–retrieval-augmented generation, CLS–classification, SUM–summarization.

leveraging LLMs to automate or assist in threat hunting, enabling blue team defenders to scale their investigations across complex threat landscapes and respond to incidents more effectively. However, despite this momentum, the application of LLMs in blue team threat hunting remains underdeveloped. Existing frameworks tend to focus on isolated analytical tasks [80, 29, 21, 11, 35], such as generating advisory recommendations without integrating earlier steps like threat group attribution. This fragmented design limits our understanding of how LLMs perform within complex, interdependent threat-hunting workflows.

To address this gap, we introduce CYBERTEAM, a practical benchmark designed to rigorously evaluate and guide the use of LLMs in blue team threat hunting. CYBERTEAM supports blue-team threat-hunting workflows through the following aspects:

Broader Coverage. CYBERTEAM is constructed from a diverse and large-scale repository of threat intelligence data sourced from 23 vulnerability databases, including MITRE series [60], NVD [64], Exploit-DB [65], VulDB [91], and CISE [18], as well as reporting platforms such as Red Hat Bugzilla [74], Oracle Security Alerts [67], and IBM X-Force [42]. This broad and representative collection is essential for capturing the complexity and variability of the modern threat landscape, enabling realistic support for both threat investigation and incident response. In addition, CYBERTEAM includes a significantly larger number of tasks and samples than existing cybersecurity benchmarks [45, 49, 6, 44], as summarized in Table 1. This extensive coverage allows for a more comprehensive and nuanced evaluation of LLM performance across a wide range of threat-hunting scenarios.

Embodied Environment. An important feature of CYBERTEAM is its structured, modular workflow for guiding LLMs within an embodied environment [95, 16]. This design is inspired by blue team practices, where analysts typically follow standardized procedures to interpret threat metadata and conduct investigations [80, 25, 14]. However, strict adherence to such procedures can limit adaptability when analyzing unstructured threat logs or addressing emerging, zero-day threats. To balance standardization and adaptability, CYBERTEAM integrates a set of embodied functions that regulate LLM behavior while allowing for open-ended reasoning where needed. As illustrated in Figure 1, CYBERTEAM first models the dependency structure among threat-hunting objectives (e.g., attribution, behavior analysis, mitigation) as a task chain, and then maps this chain into a corresponding sequence of embodied functions. In this process, functions such as NER enforce structured outputs (e.g., extracting threat actor entities), while functions like RAG support more flexible reasoning (e.g., summarizing relevant patching strategies). This design enables CYBERTEAM to constrain model behavior when precision is essential, while still allowing LLMs to engage in context-sensitive generation when analytical creativity or abstraction is required.

Evaluation Strategy. CYBERTEAM incorporates agent-based evaluation strategies tailored to each threat-hunting objective. We benchmark leading LLMs and state-of-the-art (SOTA) cybersecurity agents, comparing CYBERTEAM 's embodied function-calling approach with popular elicitation

Table 1: Comparison of cybersecurity benchmarks for LLMs.

Benchmark	Focus	#Data	#Task	#Source	Coverage	Unique Feature
CWE-Bench-Java [49]	Java vulnerability	120	4	1	Four CWE classes	Large-scale Java codes
CTIBench [6]	Cyber Threat Intelligence	2,500	3	6	CVE, CWE, CVSS, ATT&CK	Multi-choice questions (MCQ)
SevenLLM-Bench [44]	Report understanding	91,401	28	N/A	Bilingual instruction corpus	Synthetic Data, MCQ, QA
SWE-Bench [45]	Software bug fixing	2,294	12	1	GitHub issues	Python repository
CYBERTEAM (Ours)	Blue-team threat hunting	452,293	30	23	Threat-hunting lifecycle (3.1)	Open Generation, Embodied Env

strategies such as In-Context Learning (ICL) [26], Chain-of-Thought (CoT) [94], Tree-of-Thought (ToT) [96]. Our evaluation provides insights into the capabilities and limitations of current models across 30 tasks. We further analyze LLMs' capabilities in automatically identifying task dependencies, selecting appropriate embodied functions, and handling noisy threat log inputs. These experiments reveal not only underlying model limitations, such as reasoning errors and sensitivity to incomplete or ambiguous input, but also provide practical insights for blue team analysts aiming to integrate LLMs into real-world cyber defense workflows.

In summary, this paper makes the following contributions: (1) We introduce CYBERTEAM, a practice-informed, broadly scoped benchmark that enables rigorous evaluation of LLMs for blue team threat hunting, (2) we construct an embodied environment that models the dependencies among threat-hunting tasks and guides LLMs through standardized yet flexible reasoning workflow, (3) we conduct comprehensive evaluations and provide insights to improve LLM performance among threat-hunting scenarios.

2 Related Work

LLMs for Cybersecurity. Recently, LLMs have shown promise in enhancing cybersecurity tasks such as malware classification [2, 5, 70, 24, 50], code vulnerability detection [76, 51, 82], penetration testing [36, 63, 81], phishing detection [47, 34], and incident report generation [9, 85, 54]. These applications leverage the language understanding and reasoning capabilities of LLMs to analyze technical data, recommend solutions, or simulate attacker behaviors. However, existing applications typically target isolated tasks without considering broader analyst workflows. Additionally, their open-ended reasoning often results in hallucinations and inconsistencies [62, 84, 83], raising concerns about reliability in high-stakes defensive scenarios.

Cybersecurity Benchmarks. Recent benchmarks have focused on static analysis [75, 38, 12], software vulnerabilities [40, 78], and threat report generation [88, 69, 19]. These benchmarks evaluate predefined tasks such as identifying CWE categories, matching CVEs, or summarizing intelligence reports [6, 3, 13, 37]. While helpful for reproducibility, they often cover narrow domains and lack the complexity and task interdependencies inherent in real-world threat investigations. In contrast, benchmarks from other high-stakes fields (e.g., law, medicine, finance) increasingly include complex, multistep tasks requiring diverse reasoning skills [30, 93, 17, 52, 103]. Inspired by these efforts, we introduce CYBERTEAM to emphasize structured reasoning and realistic interdependencies, specifically for blue team threat hunting.

Embodied Agents. Recent research has proposed embodied agents and function-calling frameworks to structure LLM reasoning into modular, interpretable steps [28, 27, 41]. Such frameworks have achieved notable success in robotics [43, 4], database querying [46, 20], and scientific reasoning tasks [1, 89]. However, their use in cybersecurity, especially defensive operations, remains underexplored despite the need for structured workflows. Our work addresses this gap by introducing an embodied, function-guided environment aligned with blue team practices, enabling procedural reasoning within a structured analytical pipeline.

3 CYBERTEAM

In this section, we provide a detailed introduction of CYBERTEAM regarding the collected threat hunting tasks (3.1), data sources (3.2), and the embodied strategy (3.3).

Table 2: Threat hunting tasks, description of targets, corresponding embodied functions, number of instances, and evaluation metrics. We implememt 9 embodied functions: (1) NER: named entity recognition, (2) REX: regex parsing, (3) SUM: summarization, (4) SIM: text similarity matching, (5) MAP: text mapping, (6) RAG: retrieval-augmented generation, (7) SPA: text span localization, (8) CLS: classification, and (9) MATH: mathmatical calculation. We evaluate using metrics (i) F1 score, (ii) Sim: text similarity (by BERT Score [100]), (iii) Accuracy, (iv) Normalized distance between two numbers, (v) Passing rate of code execution, (vi) Hit@k ratio. Details in Appendix B and C.

Task	Analytical Target	Function	#Data	Metric
Threat Attribution				
Malware Identification	Malware delivery or toolset	NER, SUM	15,742	F1
Signature Matching	Techniques from known threat groups	NER, SIM	5,166	F1
Temporal Pattern Matching	Known work schedules	REX	4,203	Sim
Affiliation Linking	Source organizations	NER, MAP	17,583	F1
Geographic Analysis	Geographic or cultural indicators	NER, SIM	6,164	F1
Victimology Profiling	Targeted victims or attacker motives	NER, REX	18,612	F1
Infrastructure Extraction	Domains, IPs, URLs, or file hashes	NER, REX, SUM	24,129	F1
Actor Identification	The threat group or actor (e.g., APT28)	NER, RAG, MAP	17,823	F1
Campaign Correlation	Threat campaigns or incidents	NER, MAP	27,762	F1
Behavior Analysis				
File System Activity Detection	Suspicious file creation, deletion, or access	SPA, NER, SUM	4,653	Sim
Network Behavior Profiling	Patterns of external communication (e.g., C2)	SPA, NER, SUM	2,617	Sim
Credential Access Detection	Theft or misuse of credentials	SPA, NER, SUM	2,492	Sim
Execution Context Analysis	Execution behaviors by user or process	SPA, NER, SUM	23,888	Sim
Command & Script Analysis	Suspicious commands or scripts	SPA, NER, SUM	20,232	F1
Privilege Escalation Inference	Privilege escalation attempts	SPA, NER, SUM	15,953	Sim
Evasion Behavior Detection	Evasion or obfuscation techniques	SPA, NER, SUM	8,973	Sim
Event Sequence Reconstruction	Timeline of attack-related events	SUM	23,265	Sim
TTP Extraction	Tactics, techniques, and procedures	RAG, MAP	28,292	F1
Prioritization				
Attack Vector Classification	Exploitation vectors (e.g., network, local, physical)	SUM, CLS	17,448	Acc
Attack Complexity Classification	Level of hurdles required to carry out the attack	SUM, CLS	17,116	Acc
Privileges Requirement Detection	Level of access privileges an attacker needs	SUM, CLS	18,030	Acc
User Interaction Categorization	If exploitation requires user participation	SUM, CLS	17,075	Acc
Attack Scope Detection	If the vulnerability affects one/multiple components	SUM, CLS	18,570	Acc
Impact Level Classification	Consequences on confidentiality, integrity, and availability	SUM, CLS	18,736	Acc
Severity Scoring	A numerical score indicating the overall attack severity	SUM, MATH	11,507	Dist
Response & Mitigation				
Playbook Recommendation	Relevant response actions based on threat type	RAG, SUM	10,718	Hit
Security Control Adjustment	Firewall rules, EDR settings, or group policies	RAG, SUM	9,929	Sim
Patch Code Generation	Code snippets to patch the vulnerability	RAG, SUM	11,341	Pass
Patch Tool Suggestion	Security tools or utilities	RAG, SUM	9,763	Hit
Advisory Correlation	Security advisories or best practices	RAG, SUM	24,511	Hit

3.1 Threat Hunting Tasks

As shown in Table 2, CYBERTEAM reflects the full lifecycle of threat hunting tasks. Specifically, CYBERTEAM systematizes analytical tasks into four categories: **Threat Attribution**, **Behavior Analysis**, **Prioritization**, and **Response & Mitigation**. Each category captures a stage in the threat-hunting workflow from investigating cyber threats to identifying countermeasures. Specifically:

Threat Attribution aims at uncovering the origins and nature of a threat. This includes tasks such as extracting infrastructure artifacts (e.g., domains, IPs, URLs), classifying malware families based on observed behaviors, matching known threat signatures, and linking activities to known campaigns or actor groups (e.g., APT or MITRE ATT&CK [60]). Further granularity is achieved through geographic and temporal pattern analysis, as well as victimology and affiliation linking, all of which help analysts contextualize incidents in terms of their broader threat landscape.

Subsequently, **Behavior Analysis** focuses on understanding how adversaries interact with systems over time. Tasks in this category include mapping unusual file system activities, profiling network behaviors (e.g., Monitoring outbound traffic), detecting credential access, and analyzing the use of commands and scripts. Analysts aim to reconstruct sequences of attack events and associate them with specific execution contexts or behavioral patterns. The detection of techniques such as privilege escalation and defense evasion also falls within this scope. Understanding threat behaviors enhances an analyst's ability to assess the dynamics of emerging (or, zero-day) cyber threats.

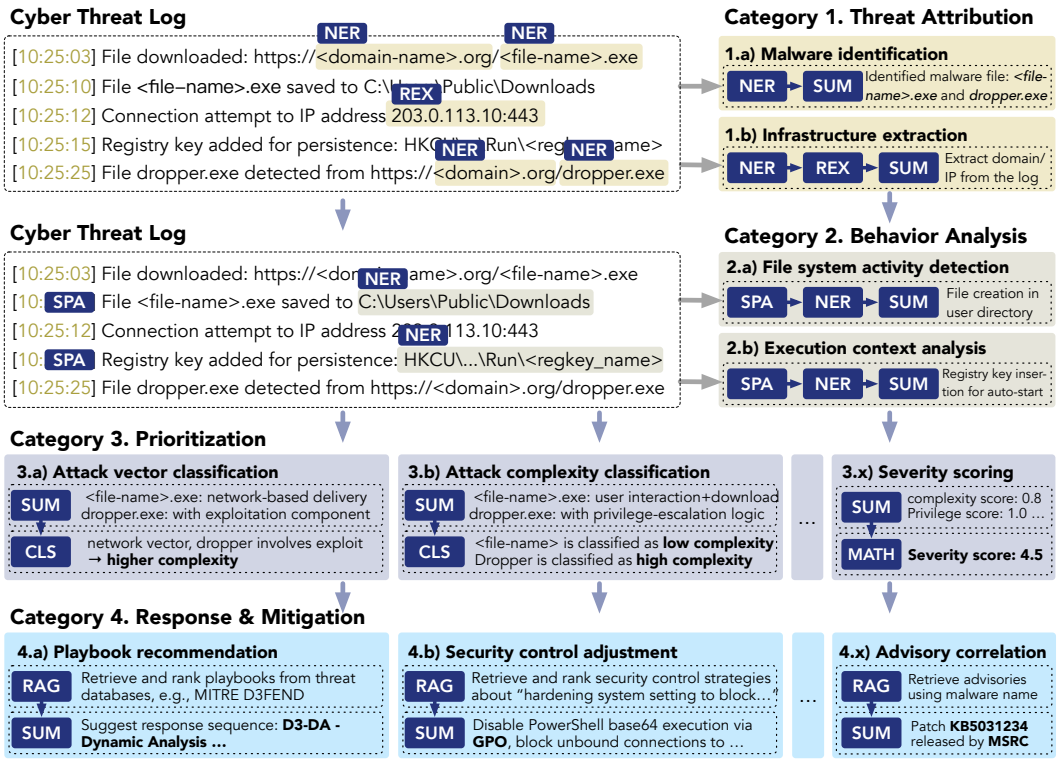

Figure 2: An threat-hunting example demonstrating a dependency chain of analytical tasks, where each task is completed through a sequence of embodied functions executed by LLMs autonomously.

When multiple threats emerge simultaneously, **Prioritization** assesses their relative urgency and associated risk. This involves analyzing the attack vector and complexity, identifying privilege requirements and user interaction dependencies, and estimating potential impact. These factors are then synthesized into impact labels and severity scores (e.g., CVSS [31]) to guide effective triage. Finally, **Response & Mitigation** focus on generating actionable defense strategies. This includes recommending response playbooks, generating patch code, correlating relevant security advisories, and suggesting appropriate tools or configuration changes to neutralize the threat.

3.2 Data Sources

CYBERTEAM collects threat metadata from two primary sources: (1) vulnerability databases, which offer authoritative structural and non-structural information about threats, and (2) threat intelligence platforms, which report event-driven, context-rich threat data.

Vulnerability databases serve as foundational resources for automated threat hunting by providing machine-readable records of software flaws, exposure types, and critical contextual metadata. We aggregate threat entries from established sources such as NVD [64], MITRE CVE [87], ATT&CK [60], CWE [58], CAPEC [57], D3FEND [59], Exploit-DB [65], and VulDB [91]. These sources include detailed insights such as exploitability scores (EPSS [32]), severity metrics (CVSS [31]), and remediation guidance. Additionally, we incorporate data from vendor-maintained repositories (e.g., the Microsoft Security Update Guide [55], IBM X-Force [42]) to capture fine-grained details on affected systems, attack vectors, and patch methods.

Threat intelligence platforms complement these databases by providing behavioral and contextual signals linked to adversary activity. Platforms such as VirusTotal [90], AlienVault OTX [7], and MISP [56] contribute indicators of compromise (IOCs), behavioral patterns, and telemetry that enable tasks like campaign correlation, infrastructure extraction, and actor attribution. Furthermore, industry threat reports—from sources, such as Mandiant [53], Recorded Future [72], Palo Alto Unit 42 [68], and Apache [86], offer semi-structured intelligence, including incident timelines, IOC lists, and

narrative analyses, which are essential for modeling multi-stage attack sequences and evaluating blue team responses.

Additional details on how these databases and platforms are used are provided in Appendix A.

3.3 Embodied Threat Hunting: Definition and Methodology

Task Dependency. Threat hunting is inherently a multi-stage analytical process [77, 15, 39], where downstream actions, such as incident response and mitigation, rely on outcomes derived from upstream analytical steps. For example, recommending an effective response playbook requires accurate attribution of the threat actor and thorough behavioral analysis of the compromise. To explicitly model this structured workflow, CYBERTEAM formulates threat hunting as a *Dependency Chain*. As illustrated in Figure 2, all analytical tasks (e.g., 1.a: Malware Identification or 2.a: File System Activity Detection) are organized into a pipelined workflow that reflects their inherent dependencies. For example, attack complexity classification relies on prior analyses of file system activity and execution context. Meanwhile, tasks within the same category (e.g., malware identification and infrastructure extraction under threat attribution) can often be performed in parallel, as they address distinct dimensions of the threat and do not exhibit direct interdependencies.

Embodied Function. Within each node, CYBERTEAM invokes a set of embodied functions designed to produce actionable threat intelligence and progressively address the current analytical target. Specifically, each threat hunting task t_i is associated with a corresponding set of tool-augmented functions $\mathcal{F}_i = \{f_i^1, f_i^2, \dots\}$. We implement a total of 9 embodied functions, and solving each task t_i involves executing a predefined sequence $f_i^* \in \mathcal{F}_i$, as detailed in the third column of Table 2. The resulting output $y_i = f_i^*(x)$ is subsequently passed to dependent downstream tasks. For instance, the task of *TTP Extraction* involves invoking both Retrieval-Augmented Generation (RAG) and Mapping (MAP) functions to identify relevant tactics and techniques from unstructured logs. Subsequently, a downstream task such as *Tool Suggestion* utilizes RAG and summarization (SUM) functions to map these identified TTPs to suitable defensive tools.

Due to space constraints, we defer implementation details embodied functions to Appendix B.

4 Experiment

CYBERTEAM aims to empirically address the following research questions: **RQ_1:** How effective are embodied functions compared to standard prompting strategies (e.g., ICL, CoT, ToT) in improving LLM performance for threat-hunting tasks? **RQ_2:** Can LLMs accurately solve individual threat-hunting tasks? **RQ_3:** Are LLMs capable of reasoning about task dependencies and selecting the appropriate embodied functions for each task? **RQ_4:** How robust are LLMs, under the guidance of CYBERTEAM, when analyzing noisy inputs?

LLMs. We evaluate a range of industry-leading large language models, including GPT-4o, GPT-o3, Qwen3-32B, Gemini-Pro, Claude-3-Opus, LLaMA-3.2-90B, LLaMA-3.1-405B, and DeepSeek-V3. In addition, we assess state-of-the-art cybersecurity-focused LLM agents, including Lily-Cybersecurity-7B [79], CyLens-8B [50], and SevenLLM-7B [44].

Elicitation (Prompting) Strategies. To compare with the embodied function approach implemented in CYBERTEAM, we evaluate three widely used prompting strategies: (1) In-Context Learning (ICL) [26] – including basic task instructions along with five illustrative examples to demonstrate the desired solution format. (2) Chain-of-Thought (CoT) [94] – encouraging the model to generate "step-by-step" reasoning results before producing the final answer. (3) Tree-of-Thought (ToT) [96] – guiding LLMs to explore multiple reasoning paths and select the most plausible one.

Metrics. As shown in Table 2, we select evaluation metrics tailored to the nature of each task. For information extraction tasks (e.g., malware identification), we use the **F1 score** to balance precision and recall. For classification tasks (e.g., privilege escalation inference), we adopt **accuracy** among well-defined categories. Generation or summarization tasks (e.g., behavioral profiling) are evaluated using **BERTScore** [101], reflecting semantic similarity. Tasks involving ranking (e.g., security playbook recommendation) utilize **Hit@k** (default $k = 10$), measuring whether correct choices appear in the top-k outputs. For programmatic outputs (e.g., patch code generation), we apply **Pass** rate using UNITEST in Python to assess functional correctness. Numeric estimation tasks (e.g.,

Table 3: Results of LLMs threat-hunting performance (scaled to 100%) on CYBERTEAM, using corresponding metrics tailored to each analytical target as detailed in Table 2. We use **boldface** indicate the best results and underline to denote the second-best results.

Model	Playbook Recommend				Security Control Adjust				Patch Code Generation				Patch Tool Suggestion				Advisory Correlation			
	ICL	CoT	ToT	Emb	ICL	CoT	ToT	Emb	ICL	CoT	ToT	Emb	ICL	CoT	ToT	Emb	ICL	CoT	ToT	Emb
Cybersecurity Agent																				
Lily-7B	42.3	51.6	48.1	**67.2**	51.5	60.3	66.7	**74.2**	10.8	24.5	25.3	**29.7**	48.2	53.6	56.5	**69.1**	21.7	49.5	46.8	**73.4**
CyLens-8B	83.5	80.6	83.8	**88.2**	75.2	80.6	83.1	**87.5**	62.4	68.7	61.9	**80.8**	74.6	80.3	83.5	**88.9**	66.5	82.1	85.3	**89.8**
SevenLLM-7B	54.7	50.5	54.3	**66.8**	43.9	68.4	61.6	**80.1**	29.2	55.1	58.3	**60.2**	61.5	77.2	68.1	**77.7**	63.8	69.5	67.2	**77.1**
Industry-Leading LLM																				
GPT-4o	64.5	78.3	75.2	**84.6**	61.8	70.3	75.9	**82.1**	56.2	58.4	61.8	**72.5**	68.9	79.2	75.8	**87.4**	64.7	67.2	70.8	**80.3**
GPT-o3	73.1	88.2	84.6	**90.2**	70.3	79.5	84.7	**88.3**	58.4	75.6	71.3	**86.9**	79.4	89.5	85.2	**96.3**	67.2	79.8	83.6	**91.7**
QWen3-32B	52.8	67.5	71.4	**79.3**	50.6	59.8	66.3	**74.7**	39.3	54.7	50.2	**65.4**	59.2	70.3	74.5	**83.6**	48.5	61.7	64.8	**76.5**
Gemini-Pro	79.4	80.1	83.5	**91.8**	65.8	79.2	73.6	**88.5**	63.7	65.3	69.8	**82.6**	74.1	81.7	86.3	**93.2**	62.4	77.5	73.1	**86.9**
Claude-Opus	63.7	80.6	76.4	**88.5**	79.2	76.3	72.1	**85.8**	47.5	65.2	60.8	**78.4**	68.5	78.3	82.9	**90.6**	56.8	75.1	71.3	**83.7**
Llama-90B	77.2	74.5	69.3	**82.6**	53.4	70.8	64.5	**79.3**	42.8	56.2	60.3	**71.5**	64.1	76.8	72.5	**85.2**	51.3	63.7	68.4	**77.8**
Llama-405B	65.8	77.3	82.1	**89.7**	61.5	77.9	72.8	**86.4**	49.2	67.4	62.9	**80.6**	70.3	79.6	84.2	**92.1**	58.7	76.3	71.8	**84.9**
DeepSeek-V3	61.4	79.8	75.1	**87.2**	57.6	74.3	68.9	**83.5**	45.7	59.8	63.5	**76.2**	67.1	76.9	81.4	**89.3**	54.2	72.8	68.4	**82.6**

severity scoring) are evaluated using **Normalized Distance** to quantify similarity to ground truth values. All metrics are scaled to the range [0, 1] (or in percentages), where higher values indicate better performance. Additional details are provided in Appendix C.

4.1 Threat-Hunting Effectiveness through Embodied Functions (RQ_1)

Ultimately, CYBERTEAM is designed to generate actionable responses and mitigation strategies against cyber threats. We begin by evaluating the overall quality of LLM-generated responses and mitigation outputs on CYBERTEAM. Table 3 presents the results, using task-specific metrics detailed in Table 2. From these results, we observe that using embodied functions outperforms standard elicitation methods. For instance, embodied functions guide GPT-o3 to achieve over 90% Hit@10 in playbook recommendation and over 91% in advisory correlation. This demonstrates the effectiveness of combining modular, task-specific guidance with the inherent flexibility of LLMs.

Notably, while ICL, CoT, and ToT have been shown to improve generation quality for general-purpose tasks [26, 98, 92], they lack meaningful guidance for domain-specific problems that require precise procedural knowledge and structured analytical workflows.

Case Study I (Failure Case). When using CoT to generate a response plan for LockBit (a ransomware), GPT-4o offers generic recommendations *"... the first step is to isolate affected machines. Next, the system should assess backup availability and notify stakeholders ..."* without tailoring to LockBit and ignoring unique traits like double extortion tactics or known exploits.

By contrast, embodied functions in CYBERTEAM constrain LLM reasoning to follow predefined task sequences, ensuring outputs remain aligned with operational goals:

Case Study II (Successful Case). The embodied function framework guides GPT-4o to explicitly invoke **RAG** and **SUM** modules. Specifically, RAG retrieves up-to-date security advisories (e.g., *CISA Alert AA23-325A*) specific to LockBit, while SUM outlines mitigation strategies with *double extortion prevention* and *air-gapped offline backups*.

These results suggest that in cybersecurity, particularly in threat-hunting scenarios, structured elicitation methods are necessary for reliably leveraging LLM capabilities.

4.2 Threat-Hunting Performance for Individual Tasks (RQ_2)

Complementing Section 4.1, we also evaluate individual threat-hunting tasks prior to the response & mitigation stage, as outlined in Table 2. Figures 3 and Appendix D present the experimental results.

Observe that using embodied functions consistently achieves the highest performance across all intermediate tasks. However, **the magnitude of performance gains varies across task types**. For instance, in complex reasoning tasks (e.g., Event Sequence Construction), embodied functions yield substantial improvements over baseline strategies like CoT and ToT, boosting accuracy by over 20% using GPT-4o. This is largely because these tasks require multi-hop reasoning, evidence synthesis,

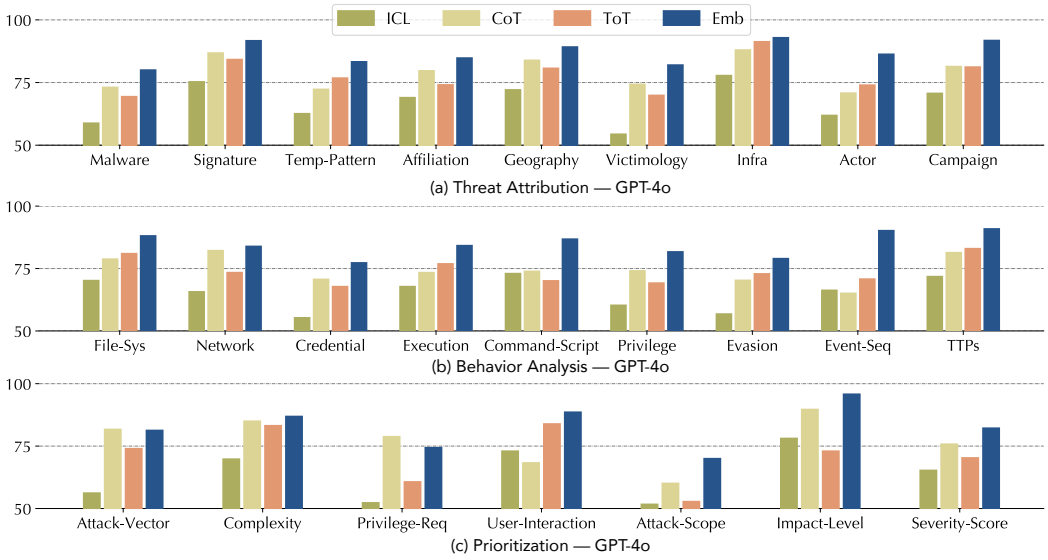

Figure 3: Threat-hunting performance (scaled to 100%) on individual tasks, evaluating under GPT-4o across various elicitation strategies: ICL, CoT, ToT, and using embodied functions (Emb). Results for additional LLMs are provided in Appendix D.

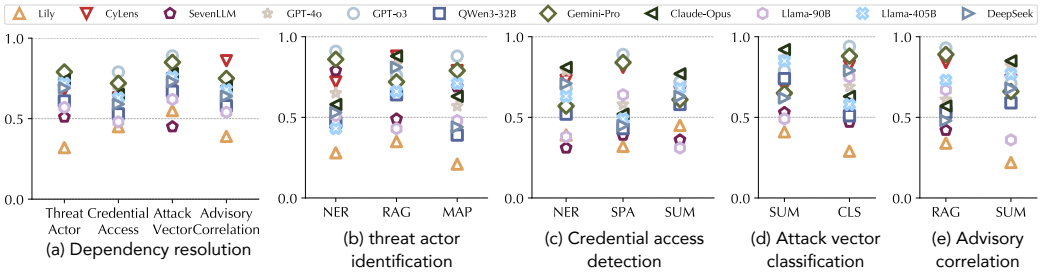

Figure 4: Evaluation of LLM performance in selecting the correct (a) task dependencies, and (b–e) embodied functions for specific analytical targets.

and careful dependency tracking, which are capabilities that general prompting methods struggle to coordinate effectively. In contrast, for narrower, classification-focused tasks (e.g., attack vector categorization or privilege escalation inference), the performance gap between embodied functions and standard prompting is smaller. Here, the tasks are more self-contained, and models can often arrive at correct predictions even without explicit task decomposition or function integration.

These findings highlight that while embodied functions offer general advantages, their relative benefit is particularly significant in scenarios requiring structured reasoning over interconnected threat-hunting steps. This demonstrates the importance of modular guidance in complex threat-hunting workflows.

4.3 Task Dependency Resolution and Embodied Function Selection (RQ_3)

Previous experiments highlight the importance of guided workflows in threat hunting. Here, we further investigate LLMs' inherent abilities to (1) identify which prior tasks provide the necessary inputs for solving a given analytical target, and (2) determine which embodied function(s) to invoke for executing specific threat-hunting tasks.

Experimental Setting. We frame both evaluations as *multi-choice problems*. For task dependency resolution, we present original threat logs from CYBERTEAM and prompt LLMs to identify which prior tasks must be resolved before proceeding with the current one. For example, when performing threat

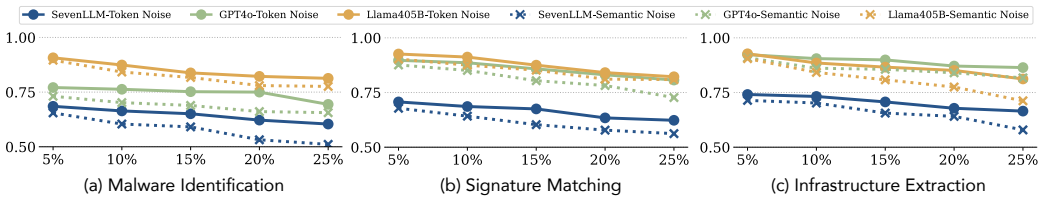

Figure 5: LLM performance when input threat logs are perturbed with token-level noise (solid line) or semantic-level noise (dashed line).

actor identification, LLMs are asked to select dependencies that provide necessary supporting evidence (e.g., signature extraction or affiliation mapping). The candidate options are drawn from the full set of tasks listed in Table 2. For embodied function selection, LLMs are asked to choose the correct function(s) required at each step of a function-calling workflow. The candidate options are drawn from the complete set of 9 embodied functions defined in our system.

We measure the **F1 score** for correct dependent task selection and **accuracy** for correct function selection at each step. Our evaluation focuses on four representative tasks spanning the end-to-end workflow, from threat attribution to response & mitigation.

Results and Observations. Figure 4 presents the evaluation results. Observe that LLMs are largely insufficient in resolving task dependencies and accurately selecting the correct embodied functions without structured guidance. Specifically, while GPT-o3 performs better in general, it only achieves 64% on identifying prior tasks required for advisory correlation. Open-source models such as Lily-7B exhibits even lower performance, often failing to recognize multi-step dependencies or indirect task linkages, leading to performance below 50%.

For embodied function selection, the results show a similar trend: Gemini-Pro and CyLens achieve average accuracy over 75%, while Llama-90B struggles to exceed 50%, particularly on complex tasks such as threat actor identification or advisory correlation. Error analysis reveals two main failure cases: (i) *over-selection*, where the model includes irrelevant functions, and (ii) *under-selection*, where the model omits necessary functions required to complete the analytical workflow.

Case Study III (Failure Case). Hallucinative Embodied Function Selection.

(i) Over-selection: In *advisory correlation*, we observe that *Llama-90B* over-selects the irrelevant *MAP* function, redundantly performing entity-to-infrastructure mapping.

(ii) Under-selection: In the *threat actor attribution*, *Claude* omits *RAG* (necessary to retrieve actor identifiers) but directly generating APT identifiers, leading to an incomplete attribution.

These observations highlight the necessity of explicitly modeling multi-step dependency resolution and modular workflows. CYBERTEAM integrates these components to substantially mitigate the identified weaknesses, enabling LLMs to navigate threat-hunting pipelines with greater reliability.

4.4 LLM Robustness against Noisy Inputs (RQ_4)

Experimental Setting. We also investigate LLM robustness when input threat logs contain noisy text. We introduce (i) token-level noise using TextAttack [61], which randomly injects or substitutes tokens, and (ii) semantic-level noise using BART-paraphraser [48], which subtly introduces misleading or shifted context. Both noise types are applied at controlled levels (e.g., perturbing 10% of the input).

Results and Observations. From Figure 5, we observe that token-level noise has a smaller impact on LLM performance compared to semantic-level noise. For example, under 10% perturbation, random character insertions or deletions lead to less than 5% performance drop across tasks. In contrast, semantic-level noise—such as paraphrased or subtly altered context—causes a much larger decline. These findings suggest that while LLMs handle surface-level errors relatively well, they struggle when the semantic shifting, even when guided by CYBERTEAM. This highlights the importance of curating expert-level threat reports in threat hunting, as imprecise statements can unintentionally mislead blue team efforts and degrade overall analysis.

5 Conclusion

We present CYBERTEAM, a benchmark designed to evaluate the capabilities of LLMs in blue team threat-hunting workflows. By combining broad and diverse real-world datasets, an embodied environment with modular function-guided reasoning, and detailed evaluation strategies, CYBERTEAM provides a comprehensive workflow for assessing LLM capabilities in realistic cyber defense scenarios. Our empirical findings highlight both the strengths and current limitations of SOTA LLMs, offering actionable insights for improving their integration into security operations. We hope CYBERTEAM will serve as a valuable resource for the research community and practitioners alike, driving future innovations in AI-assisted cybersecurity.

References

[1] Tsedeke Abate, Kassa Michael, and Carl Angell. Assessment of scientific reasoning: Development and validation of scientific reasoning assessment tool. *Eurasia Journal of Mathematics, Science and Technology Education*, 16(12):em1927, 2020.

[2] Adel Abusitta, Miles Q Li, and Benjamin CM Fung. Malware classification and composition analysis: A survey of recent developments. *Journal of Information Security and Applications*, 59:102828, 2021.

[3] Ehsan Aghaei, Waseem Shadid, and Ehab Al-Shaer. Threatzoom: Cve2cwe using hierarchical neural network. *arXiv preprint arXiv:2009.11501*, 2020.

[4] Sharath Chandra Akkaladevi, Matthias Plasch, Michael Hofmann, and Andreas Pichler. Semantic knowledge based reasoning framework for human robot collaboration. *Procedia CIRP*, 97:373–378, 2021.

[5] Jamal Al-Karaki, Muhammad Al-Zafar Khan, and Marwan Omar. Exploring llms for malware detection: Review, framework design, and countermeasure approaches. *arXiv preprint arXiv:2409.07587*, 2024.

[6] Md Tanvirul Alam, Dipkamal Bhusal, Le Nguyen, and Nidhi Rastogi. Ctibench: A benchmark for evaluating llms in cyber threat intelligence. *arXiv preprint arXiv:2406.07599*, 2024.

[7] AlienVault (AT&T Cybersecurity). Alienvault open threat exchange (otx). `https://otx.alienvault.com`, 2024.

[8] Ali Alshumrani, Nathan Clarke, and Bogdan Ghita. A unified knowledge graph to permit interoperability of heterogenous digital evidence. In *International Conference on Ubiquitous Security*, pages 420–435. Springer, 2023.

[9] Mario Luca Bernardi, Marta Cimitile, and Riccardo Pecori. Automatic job safety report generation using rag-based llms. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2024.

[10] Gavin Black, Varghese Vaidyan, and Gurcan Comert. Evaluating large language models for enhanced fuzzing: An analysis framework for llm-driven seed generation. *IEEE Access*, 2024.

[11] Andrea Borghesi, Federico Baldo, and Michela Milano. Improving deep learning models via constraint-based domain knowledge: a brief survey. *arXiv preprint arXiv:2005.10691*, 2020.

[12] Alexandre Braga, Ricardo Dahab, Nuno Antunes, Nuno Laranjeiro, and Marco Vieira. Practical evaluation of static analysis tools for cryptography: Benchmarking method and case study. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 170–181. IEEE, 2017.

[13] Ioana Branescu, Octavian Grigorescu, and Mihai Dascalu. Automated mapping of common vulnerabilities and exposures to mitre att&ck tactics. *Information*, 15(4):214, 2024.

[14] Lee Brotherston, Amanda Berlin, and William F Reyor III. *Defensive security handbook*. " O'Reilly Media, Inc.", 2024.

[15] Sergio Caltagirone, Andrew Pendergast, and Christopher Betz. The diamond model of intrusion analysis. *Threat Connect*, 298(0704):1–61, 2013.

[16] Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025.

[17] Leshem Choshen, Ariel Gera, Yotam Perlitz, Michal Shmueli-Scheuer, and Gabriel Stanovsky. Navigating the modern evaluation landscape: Considerations in benchmarks and frameworks for large language models (llms). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024): Tutorial Summaries*, pages 19–25, 2024.

[18] CISE Program. Cybersecurity information sharing environment (cise), 2024.

[19] Ondrej Čupka, Ester Federlova, and Peter Vesely. Comparison of methodologies used in cybersecurity reports. In *Developments in Information and Knowledge Management Systems for Business Applications: Volume 7*, pages 313–348. Springer, 2023.

[20] Hafsa Shareef Dar, M Ikramullah Lali, Moin Ul Din, Khalid Mahmood Malik, and Syed Ahmad Chan Bukhari. Frameworks for querying databases using natural language: a literature review. *arXiv preprint arXiv:1909.01822*, 2019.

[21] Tirtharaj Dash, Sharad Chitlangia, Aditya Ahuja, and Ashwin Srinivasan. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, 12(1):1040, 2022.

[22] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. Pentestgpt: An llm-empowered automatic penetration testing tool. *arXiv preprint arXiv:2308.06782*, 2023.

[23] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. {PentestGPT}: Evaluating and harnessing large language models for automated penetration testing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 847–864, 2024.

[24] Dharani Devadiga, Gordon Jin, Bisti Potdar, Hankyu Koo, Andrew Han, Anusha Shringi, Angad Singh, Kinjal Chaudhari, and Saurav Kumar. Gleam: Gan and llm for evasive adversarial malware. In *2023 14th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 53–58. IEEE, 2023.

[25] Yuri Diogenes and Erdal Ozkaya. *Cybersecurity-attack and defense strategies: Infrastructure security with red team and blue team tactics*. Packt Publishing Ltd, 2018.

[26] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[27] Vardhan Dongre, Xiaocheng Yang, Emre Can Acikgoz, Suvodip Dey, Gokhan Tur, and Dilek Hakkani-Tür. Respact: Harmonizing reasoning, speaking, and acting towards building large language model-based conversational ai agents. *arXiv preprint arXiv:2411.00927*, 2024.

[28] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

[29] Hossein Rajaby Faghihi, Aliakbar Nafar, Chen Zheng, Roshanak Mirzaee, Yue Zhang, Andrzej Uszok, Alexander Wan, Tanawan Premsri, Dan Roth, and Parisa Kordjamshidi. Gluecons: A generic benchmark for learning under constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9552–9561, 2023.

[30] Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Songyang Zhang, Kai Chen, Zongwen Shen, and Jidong Ge. Lawbench: Benchmarking legal knowledge of large language models. *arXiv preprint arXiv:2309.16289*, 2023.

[31] FIRST. Common vulnerability scoring system (cvss). `https://www.first.org/cvss/`.

[32] FIRST. Exploit prediction scoring system (epss). `https://www.first.org/epss/`.

[33] Angeliki Gioti. Advancements in open source intelligence (osint) techniques and the role of artificial intelligence in cyber threat intelligence (cti). Master's thesis, 2024.

[34] Francesco Greco, Giuseppe Desolda, Andrea Esposito, Alessandro Carelli, et al. David versus goliath: Can machine learning detect llm-generated text? a case study in the detection of phishing emails. In *The Italian Conference on CyberSecurity*, 2024.

[35] Tudor Groza, Tania Tudorache, Peter N Robinson, and Andreas Zankl. Capturing domain knowledge from multiple sources: the rare bone disorders use case. *Journal of Biomedical Semantics*, 6:1–15, 2015.

[36] Andreas Happe and Jürgen Cito. Getting pwn'd by ai: Penetration testing with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 2082–2086, 2023.

[37] Erik Hemberg, Jonathan Kelly, Michal Shlapentokh-Rothman, Bryn Reinstadler, Katherine Xu, Nick Rutar, and Una-May O'Reilly. Linking threat tactics, techniques, and patterns with defensive weaknesses, vulnerabilities and affected platform configurations for cyber hunting. *arXiv preprint arXiv:2010.00533*, 2020.

[38] Juan R Bermejo Higuera, Javier Bermejo Higuera, Juan A Sicilia Montalvo, Javier Cubo Villalba, and Juan José Nombela Pérez. Benchmarking approach to compare web applications static analysis tools detecting owasp top ten security vulnerabilities. *Computers, Materials & Continua*, 64(3), 2020.

[39] Caroline Hillier and Talieh Karroubi. Turning the hunted into the hunter via threat hunting: Life cycle, ecosystem, challenges and the great promise of ai. *arXiv preprint arXiv:2204.11076*, 2022.

[40] Md Imran Hossen, Jianyi Zhang, Yinzhi Cao, and Xiali Hei. Assessing cybersecurity vulnerabilities in code large language models. *arXiv preprint arXiv:2404.18567*, 2024.

[41] Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. *arXiv preprint arXiv:2408.00764*, 2024.

[42] IBM Corporation. Ibm x-force exchange, 2024.

[43] Hyeongyo Jeong, Haechan Lee, Changwon Kim, and Sungtae Shin. A survey of robot intelligence with large language models. *Applied Sciences*, 14(19):8868, 2024.

[44] Hangyuan Ji, Jian Yang, Linzheng Chai, Chaoren Wei, Liqun Yang, Yunlong Duan, Yunli Wang, Tianzhen Sun, Hongcheng Guo, Tongliang Li, et al. Sevenllm: Benchmarking, eliciting, and enhancing abilities of large language models in cyber threat intelligence. *arXiv preprint arXiv:2405.03446*, 2024.

[45] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[46] Rabiah Abdul Kadir, Ely Salwana Mat Surin, and Mahidur R Sarker. A systematic review of automated classification for simple and complex query sql on nosql database. *Computer Systems Science & Engineering*, 48(6), 2024.

[47] Aditya Kulkarni, Vivek Balachandran, Dinil Mon Divakaran, and Tamal Das. From ml to llm: Evaluating the robustness of phishing webpage detection models against adversarial attacks. *arXiv preprint arXiv:2407.20361*, 2024.

[48] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

[49] Ziyang Li, Saikat Dutta, and Mayur Naik. Iris: Llm-assisted static analysis for detecting security vulnerabilities. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] Xiaoqun Liu, Jiacheng Liang, Qiben Yan, Jiyong Jang, Sicheng Mao, Muchao Ye, Jinyuan Jia, and Zhaohan Xi. Cyber defense reinvented: Large language models as threat intelligence copilots. *arXiv preprint arXiv:2502.20791*, 2025.

[51] Guilong Lu, Xiaolin Ju, Xiang Chen, Wenlong Pei, and Zhilong Cai. Grace: Empowering llm-based software vulnerability detection with graph structure and in-context learning. *Journal of Systems and Software*, 212:112031, 2024.

[52] Mary M Lucas, Justin Yang, Jon K Pomeroy, and Christopher C Yang. Reasoning with large language models for medical question answering. *Journal of the American Medical Informatics Association*, 31(9):1964–1975, 2024.

[53] Mandiant (Google Cloud). Mandiant threat intelligence reports. `https://www.mandiant.com/resources/reports`, 2024.

[54] Sean McGregor, Allyson Ettinger, Nick Judd, Paul Albee, Liwei Jiang, Kavel Rao, William H Smith, Shayne Longpre, Avijit Ghosh, Christopher Fiorelli, et al. To err is ai: A case study informing llm flaw reporting practices. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28938–28945, 2025.

[55] Microsoft Corporation. Microsoft security update guide. `https://msrc.microsoft.com/update-guide`.

[56] MISP Project. Misp - threat intelligence sharing platform. `https://www.misp-project.org`, 2024.

[57] MITRE Corporation. Common attack pattern enumeration and classification (capec). `https://capec.mitre.org/`.

[58] MITRE Corporation. Common weakness enumeration (cwe). `https://cwe.mitre.org/`.

[59] MITRE Corporation. D3fend: A knowledge graph of cybersecurity countermeasures. `https://d3fend.mitre.org/`.

[60] MITRE Corporation. Mitre att&ck framework, 2024.

[61] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, 2020.

[62] Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*, 2023.

[63] Lajos Muzsai, David Imolai, and András Lukács. Hacksynth: Llm agent and evaluation framework for autonomous penetration testing. *arXiv preprint arXiv:2412.01778*, 2024.

[64] National Institute of Standards and Technology (NIST). National vulnerability database (nvd), 2024.

[65] Offensive Security. Exploit database (exploit-db), 2024.

[66] Yaroslav Oliinyk, Michael Scott, Ryan Tsang, Chongzhou Fang, Houman Homayoun, et al. Fuzzing {BusyBox}: Leveraging {LLM} and crash reuse for embedded bug unearthing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 883–900, 2024.

[67] Oracle Corporation. Oracle security alerts, 2024.

[68] Palo Alto Networks. Unit 42 threat research reports. `https://unit42.paloaltonetworks.com`, 2024.

[69] Filippo Perrina, Francesco Marchiori, Mauro Conti, and Nino Vincenzo Verde. Agir: Automating cyber threat intelligence reporting with natural language generation. In *2023 IEEE International Conference on Big Data (BigData)*, pages 3053–3062. IEEE, 2023.

[70] Xingzhi Qian, Xinran Zheng, Yiling He, Shuo Yang, and Lorenzo Cavallaro. Lamd: Context-driven android malware detection and classification with llms. *arXiv preprint arXiv:2502.13055*, 2025.

[71] Jeyavijayan Rajendran, Vinayaka Jyothi, and Ramesh Karri. Blue team red team approach to hardware trust assessment. In *2011 IEEE 29th international conference on computer design (ICCD)*, pages 285–288. IEEE, 2011.

[72] Recorded Future. Recorded future threat intelligence reports. `https://www.recordedfuture.com/research`, 2024.

[73] Red Hat, Inc. Red hat security advisories (rhsa). `https://access.redhat.com/`.

[74] Red Hat, Inc. Red hat bugzilla, 2024.

[75] Ann Marie Reinhold, Brittany Boles, A Redempta Manzi Muneza, Thomas McElroy, and Clemente Izurieta. Surmounting challenges in aggregating results from static analysis tools. *Military Cyber Affairs*, 7(1):5–11, 2024.

[76] Rebecca Russell, Louis Kim, Lei Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul Ellingwood, and Marc McConley. Automated vulnerability detection in source code using deep representation learning. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 757–762. IEEE, 2018.

[77] Clemens Sauerwein, Christian Sillaber, Andrea Mussmann, and Ruth Breu. A framework for cyber threat hunting. In *ACM CCS Workshop on Security and Privacy Analytics*, 2019.

[78] Devesh Sawant, Manjesh K Hanawal, and Atul Kabra. Improving discovery of known software vulnerability for enhanced cybersecurity. *arXiv preprint arXiv:2412.16607*, 2024.

[79] Segolily Labs. Lily-Cybersecurity-7B-v0.2. `https://huggingface.co/segolilylabs/Lily-Cybersecurity-7B-v0.2`, 2025.

[80] Kunal Sehgal and Nikolaos Thymianis. *Cybersecurity Blue Team Strategies: Uncover the secrets of blue teams to combat cyber threats in your organization*. Packt Publishing Ltd, 2023.

[81] Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, and Wei Ruan. Pentestagent: Incorporating llm agents to automated penetration testing. *arXiv preprint arXiv:2411.05185*, 2024.

[82] Ze Sheng, Fenghua Wu, Xiangwu Zuo, Chao Li, Yuxin Qiao, and Lei Hang. Lprotector: An llm-driven vulnerability detection system. *arXiv preprint arXiv:2411.06493*, 2024.

[83] Aryan Shrivastava. Response inconsistency of large language models in high-stakes military decision making.

[84] Adi Simhi, Itay Itzhak, Fazl Barez, Gabriel Stanovsky, and Yonatan Belinkov. Trust me, i'm wrong: High-certainty hallucinations in llms. *arXiv preprint arXiv:2502.12964*, 2025.

[85] Fahim Sufi. An innovative gpt-based open-source intelligence using historical cyber incident reports. *Natural Language Processing Journal*, 7:100074, 2024.

[86] The Apache Software Foundation. Apache security advisories. `https://www.apache.org/security/`, 2024.

[87] The MITRE Corporation. Common Vulnerabilities and Exposures (CVE). `https://cve.mitre.org/`, n.d.

[88] Norbert Tihanyi, Mohamed Amine Ferrag, Ridhi Jain, Tamas Bisztray, and Merouane Debbah. Cybermetric: a benchmark dataset based on retrieval-augmented generation for evaluating llms in cybersecurity knowledge. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 296–302. IEEE, 2024.

[89] Krist Vaesen and Wybo Houkes. A new framework for teaching scientific reasoning to students from application-oriented sciences. *European journal for philosophy of science*, 11(2):56, 2021.

[90] VirusTotal (Google Chronicle). Virustotal: Analyze suspicious files and urls. `https://www.virustotal.com`, 2024.

[91] VulDB Team. Vuldb vulnerability database, 2024.

[92] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022.

[93] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[94] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[95] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16227–16237, 2024.

[96] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[97] Ting Yu, Simeon Simoff, and Tony Jan. Vqsvm: a case study for incorporating prior domain knowledge into inductive machine learning. *Neurocomputing*, 73(13-15):2614–2623, 2010.

[98] Zihan Yu, Liang He, Zhen Wu, Xinyu Dai, and Jiajun Chen. Towards better chain-of-thought prompting strategies: A survey. *arXiv preprint arXiv:2310.04959*, 2023.

[99] Jie Zhang, Haoyu Bu, Hui Wen, Yongji Liu, Haiqiang Fei, Rongrong Xi, Lun Li, Yun Yang, Hongsong Zhu, and Dan Meng. When llms meet cybersecurity: A systematic literature review. *Cybersecurity*, 8(1):1–41, 2025.

[100] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations (ICLR)*, 2020. Official implementation: `https://github.com/Tiiiger/bert_score`.

[101] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.

[102] Qunzhi Zhou, Yogesh Simmhan, and Viktor Prasanna. Knowledge-infused and consistent complex event processing over real-time and persistent streams. *Future Generation Computer Systems*, 76:391–406, 2017.

[103] Yuxuan Zhou, Xien Liu, Chen Ning, Xiao Zhang, Chenwei Yan, Xiangling Fu, and Ji Wu. Revisiting the scaling effects of llms on medical reasoning capabilities.

A Data Source and Metadata Collection

The MITRE CVE (Common Vulnerabilities and Exposures) system [87] is a foundational database that provides unique identifiers for publicly disclosed cybersecurity vulnerabilities. Each CVE record includes an ID, a brief description, references to external resources, and associated vendors or platforms. This source allows for consistent naming and indexing of vulnerabilities across tools and reports. We collect structured metadata such as CVE IDs, descriptions, reference links, and related CWE classifications. CVE feeds (XML/JSON) are used for automated ingestion and linkage to other threat intelligence frameworks like CAPEC and ATT&CK.

Maintained by NIST, the **NVD (National Vulnerability Database)** [64] builds on MITRE CVE data by adding rich metadata, including CVSS scores (base, temporal, environmental), CWE mappings, configuration impacts, patch availability, and severity vectors. We extract metadata through the official JSON data feeds, parsing CVE-level risk metrics, impact sub-scores, and associated product configurations. This information is critical for prioritizing remediation and understanding the real-world impact of vulnerabilities.

Exploit-DB [65] is a curated collection of publicly available exploits and proof-of-concept code. Each entry includes exploit titles, CVE references, author information, platform tags, and the actual code used in attacks. Unlike CVE/NVD, Exploit-DB provides practical insights into how vulnerabilities are weaponized in real environments. We extract titles, descriptions, exploit types (e.g., Local, Remote), and related CVEs using web scraping and NLP-based text classification.

CWE(Common Weakness Enumeration) [58] is a taxonomy developed by MITRE to classify software and hardware weaknesses. Each CWE includes a unique ID, a detailed explanation, potential consequences, examples, and related patterns (e.g., CAPEC). We use CWE to enrich CVE data with root cause information, enabling fine-grained vulnerability clustering and defensive prioritization. Metadata includes weakness category, severity, and relationships with CAPEC and CVE entries.

CAPEC (Common Attack Pattern Enumeration and Classification) [57] provides a standardized catalog of common attack strategies. Each pattern includes the attacker's objectives, prerequisites, execution flow, related weaknesses (CWE), and example scenarios. We extract attack pattern IDs, descriptions, related CWEs, and suggested mitigations. These data points enable us to map vulnerabilities to adversarial behaviors, enhancing our CTI behavioral modeling capabilities.

The MITRE ATT&CK [60] framework systematically catalogs adversary tactics, techniques, and procedures (TTPs) observed in real-world incidents. Each entry includes tactic categories (e.g., Privilege Escalation), techniques, mitigations, detection suggestions, and threat actor mappings. We extract technique IDs, corresponding software, mitigation strategies, and detection methods. These are used to link CVEs and exploits to higher-level attacker behaviors, supporting advanced threat modeling.

D3FEND [59] is a curated knowledge graph that maps defensive techniques to specific threat behaviors and artifacts. D3FEND complements the well-known ATT&CK framework by focusing on how defenders can detect, disrupt, and respond to adversarial actions. To integrate this resource into CYBERTEAM, we crawl D3FEND's publicly available ontology and extract metadata on detection, deception, and mitigation techniques, along with their associated digital artifacts (e.g., file paths, registry keys, network signatures). This metadata is then linked to relevant analytical tasks—such as behavioral profiling and response planning—providing a rich, standardized reference for grounding LLM outputs in practical defensive actions.

Oracle Security Alerts [67] provides detailed security patch advisories for its product suite. Each alert includes the CVEs addressed, severity scores, and remediation timelines. We parse the advisories to gather product-specific vulnerability timelines, vendor patch statuses, and mitigation instructions, which complement the NVD and MITRE CVE datasets.

Red Hat Bugzilla [74] is a bug tracking system that includes detailed discussions and technical logs about software bugs, many of which are security-related. Entries often include CVE links, fix status, patch availability, and affected components. We scrape metadata such as Bug IDs, CVE references, affected packages, and resolution details to supplement our understanding of vulnerability lifecycle management.

The RHSA(Red Hat Security Advisories) [73] portal lists all critical, important, and moderate security advisories affecting Red Hat products. Each advisory provides CVE mappings, severity scores, fixed packages, and risk summaries. Metadata extraction includes advisory IDs, publication dates, CVE linkages, and suggested upgrades or patches, enabling alignment with real-world remediation practices.

IBM X-Force Exchange [42] is a commercial threat intelligence sharing platform that provides in-depth reports on vulnerabilities, exploits, malware, and threat actors. Each CVE entry is enriched with exploitability status, malware connections, and actor attribution. We extract structured threat metadata such as exploit availability, indicators of compromise (IOCs), campaign tags, and actor profiling to complement CVE risk modeling.

CISE (Cybersecurity Information Sharing Environment) [18], maintained by CISA, promotes cybersecurity information exchange across government and private sector entities. The platform facilitates sharing of indicators of compromise (IOCs), analysis reports, and threat mitigation strategies through structured partnerships. We extract strategic-level threat metadata, including threat vectors, vulnerability trends, and response best practices from shared reports and alerts. This supports broader CTI tasks like attribution and risk contextualization.

VulDB (Vulnerability Database) [91] is a commercial vulnerability intelligence service that provides insights into current exploits, threat actor behavior, and exploit trends. Entries often include exploitability scores, attack vectors, exploitation status, and tags related to malware or campaigns. We collect CVE mappings, vulnerability titles, exploitation timelines, and associated actors, enabling temporal and behavioral correlation with other sources like Exploit-DB and MITRE ATT&CK .

Apache's official security advisory page lists all disclosed vulnerabilities affecting Apache projects (e.g., HTTP Server, Tomcat, Struts) [86]. Each advisory includes CVE references, affected versions, and patch instructions. We extract CVE mappings, patch details, vulnerability types, and affected modules. These insights are cross-referenced with MITRE CVE and NVD entries to improve accuracy in software-specific threat tracking.

Mandiant Threat Intelligence Reports [53], now part of Google Cloud, publishes in-depth research on nation-state APTs, malware campaigns, and threat actor tactics. Their reports include IOC lists, ATT&CK mappings, and campaign chronologies. We extract metadata on APT groups, attack stages, observed TTPs, and malware toolkits. These data points support the attribution and behavioral modeling dimensions of our threat intelligence corpus.

Recorded Future Threat Intelligence Reports [72] publishes real-time, machine-readable threat intelligence covering threat actors, vulnerabilities, dark web chatter, and geopolitical cyber campaigns. Reports often include structured indicators, predictive analytics, and CVE exploitability assessments. We leverage this source to collect threat context, emerging trends, and exploit discussion patterns—enabling our system to associate vulnerabilities with evolving threat actor intent and capability .

Unit 42 Threat Research (Palo Alto Networks) [68] provides malware analysis, campaign forensics, and actor behavior insights from Palo Alto Networks' global threat intelligence platform. Their publications include links to malicious infrastructure, malware families, and ATT&CK references. We extract TTPs, CVE-to-malware correlations, and campaign data. This enhances our contextual metadata for linking specific vulnerabilities to real-world exploitation scenarios .

Microsoft's Security Update Guide [55] lists monthly updates across its software stack. Entries contain CVEs, severity ratings, exploitability assessments, patch availability, and affected platforms. Metadata extraction includes CVE linkage, threat vectors (e.g., local, remote), exploitation likelihood, and patch rollout status—enriching vendor-specific vulnerability intelligence .

CVSS (Common Vulnerability Scoring System) [31] is a widely adopted scoring system developed by FIRST to assess the severity of software vulnerabilities. It breaks down risk into Base, Temporal, and Environmental components. We use this framework to interpret NVD scores, compare severity across platforms, and calibrate exploitability in relation to business-critical systems.

EPSS (Exploit Prediction Scoring System) [32], also developed by FIRST, provides probabilistic predictions of whether a vulnerability is likely to be exploited in the wild. It integrates data from CVSS, Exploit-DB, and historical attack patterns. We ingest EPSS scores via API to prioritize

vulnerabilities not just by severity, but by real-world exploitation likelihood—enabling dynamic risk-based vulnerability management.

MISP (Malware Information Sharing Platform) [56] is an open-source platform designed for structured threat intelligence sharing using STIX/TAXII formats. It facilitates sharing of IOCs, threat event correlations, and TTP mappings. We integrate MISP data via its API to ingest indicators (e.g., hashes, domains, IPs), related threat actors, and event metadata. These enrich our knowledge graph with actionable CTI feeds.

VirusTotal [90] is a widely used threat intelligence platform that aggregates malware analysis and sandbox reports from multiple antivirus engines and security vendors. To support behavior analysis and attribution tasks, CYBERTEAM collects structured threat metadata from VirusTotal's public API, including file hashes (MD5, SHA-1, SHA-256), behavioral execution traces, contacted IPs/domains, dropped files, and detection labels. This information is linked to threat artifacts such as malware families, indicators of compromise (IOCs), and known campaign signatures. The extracted metadata enables CYBERTEAM to contextualize adversarial behaviors and enrich analytical functions like malware classification, infrastructure extraction, and campaign correlation.

AlienVault Open Threat Exchange (OTX) [7] is a collaborative threat-sharing platform that provides community-contributed threat indicators and contextual threat intelligence. CYBERTEAM leverages the OTX API to collect threat pulses—curated collections of IOCs and metadata describing specific threat actors, campaigns, or vulnerabilities. These pulses include information such as associated IPs, domains, file hashes, CVEs, and targeted sectors. By integrating OTX data, CYBERTEAM enhances its ability to support tasks like actor attribution, TTP matching, and community correlation, allowing LLMs to reason over shared intelligence and align analysis with ongoing threat landscapes.

B Embodied Functions: Complementary Detail

To support modular and extensible capabilities within our CYBERTEAM, we decompose complex NLP workflows into discrete, embodied functions. This section detail the implementation of twelve NLP modules as described in section 3.1. Each function corresponds to a specific operation type, described as follows:

B.1 NER (Named Entity Recognition)

To identify and classify cybersecurity-relevant entities such as threat actors, malware names, vulnerabilities, and indicators of compromise (IOCs) in unstructured textual data, NER facilitates automated extraction for threat attribution and situational awareness. We employ prompt-based techniques that enable entity recognition without retraining, thus maintaining adaptability to emerging domain vocabulary.

Prompt 1. NER Prompt for Threat Attribution

System Prompt: You are a cybersecurity threat intelligence assistant specialized in named entity recognition. Your task is to extract and categorize all named entities relevant to threat attribution from the provided text. Focus on answering: *"Who is responsible for the attack?"*, *"How was the attack carried out?"*.
Instructions: Given a cybersecurity-related document or report excerpt, extract all relevant named entities and classify them into:

- **Threat Actor:** Individual(s) or groups suspected or known to conduct the activity.
- **Malware/Tool:** Names of malicious software, exploits, or hacking tools.
- **Vulnerability:** CVE identifiers or technical flaws exploited.
- **Infrastructure:** IPs, domains, file hashes, or URLs used.

Output: Return results as a structured JSON object.

B.2 REX (Regex Parsing)

To extract structured indicators from cybersecurity logs or reports, REX employs predefined regular expressions to match patterns like IP addresses, domain names, file hashes, and timestamps. This rule-based approach offers high precision in normalizing threat data for correlation and enrichment tasks.

Prompt 2. Regex Pattern Matching Prompt

System Prompt: You are a cybersecurity parsing assistant. Your task is to extract standard threat indicators from raw incident reports using predefined regex patterns.
Instructions: Parse the following document and extract any matches for:

- IP addresses
- File hashes (MD5, SHA1, SHA256)
- Domain names
- Timestamps

Output: Return all matches grouped by type in structured JSON format.

B.3 SUM (Summarization)

To enable analysts to quickly grasp key information from lengthy threat reports, SUM generates concise summaries while preserving critical details such as TTPs, IOCs, and incident timelines.

Prompt 3. Threat Report Summarization Prompt

System Prompt: You are a cybersecurity analyst assistant. Your task is to summarize the following threat report in 3–4 sentences, preserving the attack vector, affected systems, timeline, and any mentioned threat actors or IOCs.
Instructions: Summarize only the essential intelligence. Avoid generic phrases. Include dates, names, and tools where available.
Output: Return a plain-text summary paragraph.

B.4 SIM (Text Similarity Matching)

To determine semantic equivalence between pairs of threat indicators—particularly geographic or cultural references (e.g., "Eastern European" vs. "Russian-speaking")—the SIM function applies LLM-based textual similarity matching. This is critical for normalizing contextual descriptions found in incident reports or threat assessments that use varied, informal, or aliasing terms to describe similar threat origin profiles. Rather than relying on surface-level keyword overlap, SIM leverages the LLM's contextual understanding to judge whether two descriptions refer to the same underlying group or region. This helps unify disparate threat intelligence entries that may use different terminology for the same adversarial origin.

Prompt 4. Text Similarity Matching Prompt for Geocultural Indicators

System Prompt: You are a cybersecurity assistant that helps analysts determine whether two geolocation or cultural indicators refer to the same threat origin. Use contextual reasoning to decide if the two phrases describe the same group or region in a cyber threat context.
Instructions: Given two input phrases describing threat origin (e.g., "Russian-affiliated" vs. "Eastern Bloc actor"), determine whether they semantically refer to the same group or geopolitical background.
Answer the following questions:

- Do both descriptions point to the same cultural, linguistic, or geopolitical region?
- Are the expressions used interchangeably in threat intelligence contexts?

Output: Return a JSON object with:

- `"match"`: Boolean (true/false)
- `"confidence"`: A float score from 0.0 to 1.0
- `"justification"`: One or two sentences explaining the decision

B.5 MAP (Text Mapping)

To visualize and semantically relate named entities and key concepts extracted from cybersecurity documents, the **MAP** function supports construction of structured representations such as knowledge graphs or threat maps. These representations help uncover infrastructure relationships, campaign patterns, and geotemporal dynamics in threat activity. When powered by large language models, MAP enables flexible and context-aware extraction of relational triples from unstructured threat reports.

Prompt 7. Threat Knowledge Mapping Prompt

System Prompt: You are a cybersecurity knowledge graph assistant. Extract and relate key entities from the given threat report to form subject-predicate-object triples.
Instructions: Identify entities (e.g., threat actors, tools, organizations, IP addresses) and the relationships between them (e.g., "uses", "targets", "associated with").
Output: Return a list of triples in the format: `[subject, predicate, object]` Include a confidence score (0–1) if applicable.

B.6 RAG (Retrieval-Augmented Generation)

To enhance generation with accurate and recent data, RAG combines LLM output with real-time retrieval from external threat intelligence APIs or databases. It is particularly useful for describing evolving threats or identifying actor affiliations.

Prompt 4. Structured Query for Retrieval

System Prompt: You are a cybersecurity assistant. Formulate a concise search query to retrieve current information about the topic specified below.
Instructions: Based on the topic *"Recent activity by APT29 involving phishing attacks"*, generate a query such as:
"APT29 phishing campaign 2024 indicators, tools, and targets site:mitre.org OR site:virustotal.com"
Output: Return the final query string and optionally list key evidence passages from results.

B.7 SPA (Text Span Localization)

To precisely extract actionable phrases—such as indicators of compromise or technique descriptions—from long-form cybersecurity text, **Text Span Localization** (SPA) models are used.

Two key metrics evaluate SPA effectiveness:

- **Exact Match (EM)**:

$$\text{EM} = \frac{\text{Number of exact matches}}{\text{Total predictions}}$$

- **Intersection over Union (IoU)**:

$$\text{IoU} = \frac{|S_p \cap S_t|}{|S_p \cup S_t|}$$

These metrics assess both strict and partial correctness, aiding in accurate downstream processing such as relation extraction or automated summarization.

Prompt 5. Span Extraction Prompt

System Prompt: You are a cybersecurity span identification assistant. Extract the text span that describes the primary technique used in the attack.
Instructions: Given a report excerpt, locate and return the sentence or phrase that directly describes how the attacker compromised the system (e.g., phishing, lateral movement, privilege escalation).
Output: Return the extracted span as plain text.

B.8 CLS (Classification)

To measure the ability of a system to categorize cybersecurity-relevant textual inputs—such as threat alerts, vulnerability descriptions, or log messages—into predefined classes (e.g., threat categories, severity levels, or attack types), classification models are employed. This is commonly performed using transformer-based large language models (LLMs), which utilize a special token (e.g., [CLS]) to represent sentence-level semantics. The resulting embedding is mapped to labels through a learned classifier.

B.9 MATH (Mathematical Calculation)

To perform quantitative analyses and structured computations relevant to cybersecurity, the **MATH** function supports tasks such as frequency modeling, impact scoring, cryptographic evaluation, and automated threat prioritization. These computations are critical for risk-informed decision-making within cyber threat intelligence pipelines.

A prominent example is the **Common Vulnerability Scoring System (CVSS v3.1)**, which uses a combination of weighted factors and conditional logic to produce a standardized severity score for vulnerabilities. One key element is the *Base Score*, calculated using the Impact and Exploitability sub scores:

$$\text{Base Score} = \begin{cases} 0, & \text{if Impact Subscore } \leq 0 \\ \text{RoundUp}\left(\min(\text{Impact} + \text{Exploitability}, 10)\right), & \text{if Scope is Unchanged} \\ \text{RoundUp}\left(\min(1.08 \times (\text{Impact} + \text{Exploitability}), 10)\right), & \text{if Scope is Changed} \end{cases}$$

The *Impact Subscore* is computed from confidentiality, integrity, and availability impact metrics as:

$$\text{ISC}_{\text{Base}} = 1 - (1 - C) \times (1 - I) \times (1 - A)$$

This formula models the probability that the system's security properties are affected by a vulnerability. The resulting score guides patching priority, risk exposure assessments, and automated vulnerability triage.

Such logic-heavy, non-trivial calculations exemplify the role of mathematical modules in operational cybersecurity settings and justify the integration of computational reasoning capabilities in modern cyber AI systems.

Prompt 9. CVSS Score Computation Prompt

System Prompt: You are a cybersecurity scoring assistant. Given a vulnerability description and metric values (Confidentiality, Integrity, Availability, Scope, Attack Vector, etc.), compute the CVSS v3.1 Base Score.
Instructions: Use the official CVSS equations and apply the rounding rules specified in the standard. Return both the numeric score and a textual explanation of the computation steps.
Output: Return the Base Score as a float (1 decimal place) and a step-by-step explanation.

C Metric

C.1 Sim (BERT Score)

To evaluate the semantic similarity between cybersecurity-related texts—such as comparing analyst-written threat summaries, aligning generated incident narratives with original reports, or verifying paraphrased explanations of threat indicators—the **Sim** function utilizes contextual embedding-based metrics. Specifically, it computes **BERTScore** [100], which has been shown to correlate strongly with human judgment in natural language generation tasks.

BERTScore measures semantic equivalence at the token level by aligning contextual embeddings from pre-trained transformer models. The score is computed as:

$$\text{BERTScore} = \frac{1}{|x|} \sum_{i} \max_{j} \cos(\mathbf{x_i}, \mathbf{y_j})$$

where $\mathbf{x_i}$ and $\mathbf{y_j}$ are contextual embeddings of tokens in the candidate and reference texts, respectively. The final score reflects the average of maximal cosine similarities for each token in the candidate sentence.

This metric is particularly valuable in evaluating machine-generated text in cybersecurity domains, where surface-level similarity may fail to capture the deeper equivalence of technical meaning or threat context.

C.2 Pass (Code Execution Passing Rate)

To measure the reliability and functional correctness of cybersecurity automation artifacts—such as detection rules, analysis scripts, or integration workflows—the **Pass Rate** metric is employed. It quantifies how well a system performs under test by evaluating the proportion of test cases that execute successfully within a defined execution cycle, often conducted in a continuous integration (CI) pipeline.

Formally, the Pass Rate is defined as:

$$\text{Pass Rate} = \frac{\text{Number of Passed Tests}}{\text{Total Tests Executed}} \times 100\%$$

This metric provides a coarse yet effective indicator of operational readiness. A high Pass Rate implies that the deployed codebase functions as intended across its tested scenarios, which is critical in cybersecurity contexts where automation is used to process threat intelligence, detect anomalies, or trigger incident response mechanisms.

Routine monitoring of this metric supports the early identification of integration regressions, promotes pipeline stability, and ensures confidence in deploying automated defensive measures to production environments.

C.3 Hit (Top-k Hit Ratio)

To evaluate the effectiveness of cybersecurity recommendation or retrieval systems—such as those that propose relevant threat indicators, patch suggestions, attack techniques, or investigative leads—the

Top-k Hit Ratio is employed. This metric measures how frequently at least one correct or relevant item appears within the top-k ranked results returned by the system.

Mathematically, the Top-k Hit Ratio is defined as:

$$\text{Hit@}k = \frac{\text{Number of queries with at least one relevant item in top } k}{\text{Total number of queries}}$$

A higher Hit@k indicates better system performance in surfacing relevant intelligence near the top of recommendations, which is critical for time-sensitive security operations.

Use Case Example: If a system recommends threat indicators based on a query about a ransomware family, Hit@5 evaluates whether at least one valid IOC (e.g., file hash or C2 domain) appears in the top 5 returned items.

Prompt 6. Hit Evaluation Prompt for Threat Retrieval

System Prompt: You are an assistant for evaluating cybersecurity retrieval systems. Given a query and a list of system-generated recommendations, check whether any ground truth item appears within the top-k returned results.
Instructions: For each query, compare the top-k predicted items against the gold-standard set. Indicate `"Hit"` if at least one match exists, otherwise `"Miss"`.
Output: Return a JSON object with fields: `query`, `top_k_results`, `ground_truth`, `hit@k`: true/false

C.4 Dist (Normalized Distance Similarity)

To evaluate the accuracy of numeric predictions in range-based estimation tasks, such as severity scoring, the **Normalized Distance Similarity** (**Dist**) metric is employed. This metric compares the predicted number and the ground-truth and scales the similarity into the $[0, 1]$ range, where higher values indicate closer alignment.

Formally, the similarity is computed as:

$$\text{Similarity} = 1 - \frac{|\hat{c} - c|}{R}$$

where $\hat{c}$ and c denote the midpoints of the predicted and true ranges, respectively, and R is the maximum possible value of the range (e.g., 10 in our case of CVSS scores). The metric reflects the Euclidean distance between prediction and truth, normalized such that a perfect match yields a similarity of 1, and the furthest possible discrepancy yields 0.

D Additional Experimental Results

This section presents additional experimental results that complement our main findings, offering deeper insights into model behavior across varied threat-hunting scenarios.

Individual Threat Hunting Performance. Figure 6, 7, and 8 complement the results as present in Figure 3, offering aligned insights as exhibited in previous experiments.

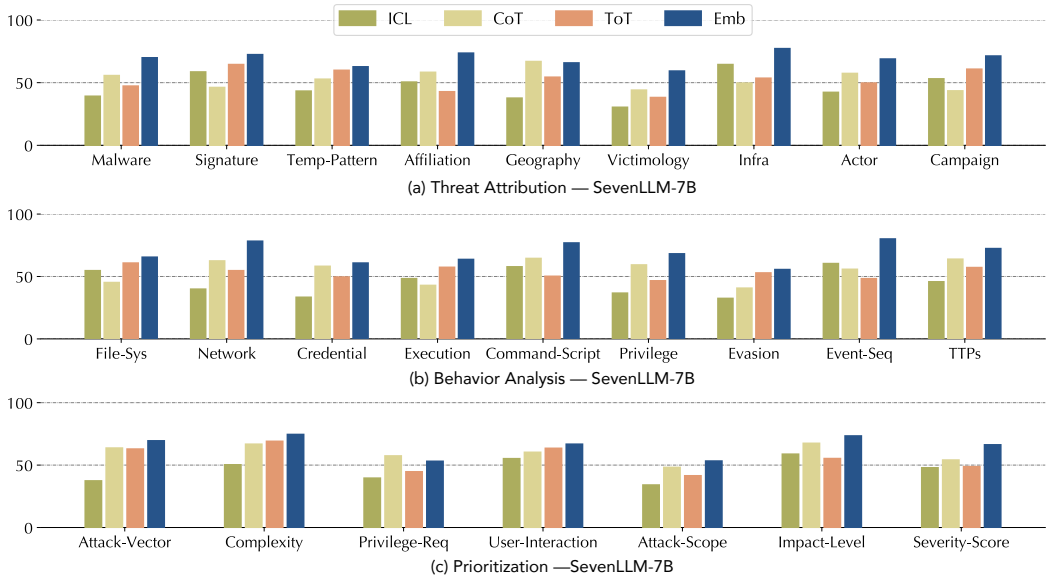

Figure 6: Threat-hunting performance on individual tasks, evaluating under SevenLLM-7B.

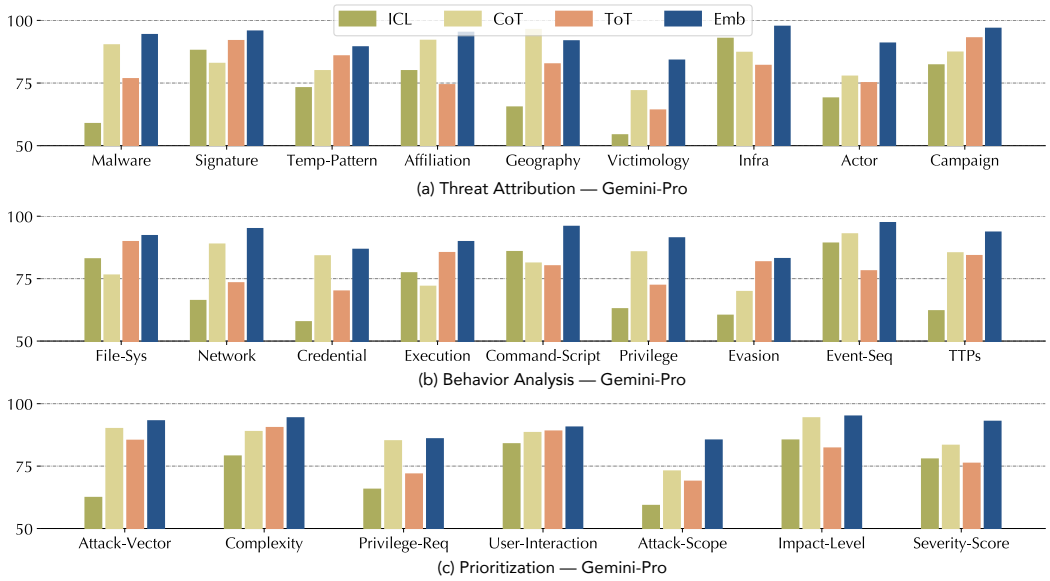

Figure 7: Threat-hunting performance on individual tasks, evaluating under Gemini-pro.

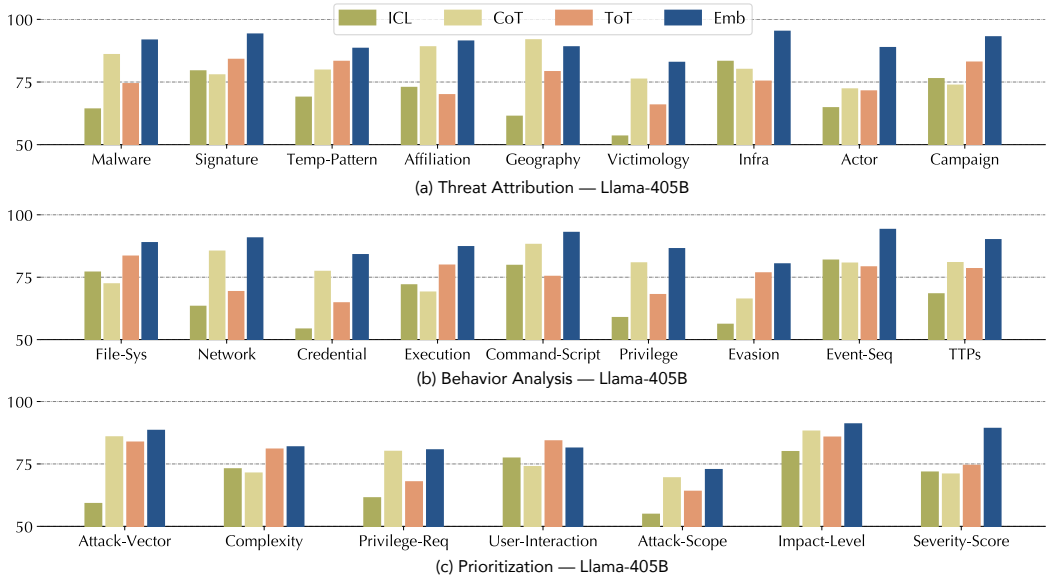

Figure 8: Threat-hunting performance on individual tasks, evaluating under Llama-405B.