BENCHMARKING LLM-ASSISTED BLUE TEAMING VIA STANDARDIZED THREAT HUNTING

Anonymous authors
Paper under double-blind review

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. This paper presents CYBERTEAM, a benchmark designed to guide LLMs in blue teaming practice. CYBERTEAM constructs a standardized workflow 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 operational modules tailored to its specific analytical requirements. This transforms threat hunting into a structured sequence of reasoning steps, with each step grounded in a discrete operation and ordered according to task-specific dependencies. Guided by this framework, LLMs are directed to perform threat-hunting tasks through modularized steps. Overall, CYBERTEAM integrates 30 tasks and 9 operational modules to guide LLMs through standardized threat analysis. We evaluate both leading LLMs and state-of-the-art cybersecurity agents, comparing CYBERTEAM against open-ended reasoning strategies. Our results highlight the improvements enabled by standardized design, while also revealing the limitations of open-ended reasoning in real-world threat hunting.

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 (The MITRE Corporation, n.d.). Defenders, commonly known as the **blue team** (Diogenes & Ozkaya, 2018; Rajendran et al., 2011), are under increasing pressure to identify, analyze, and respond to malicious activities in a timely and accurate manner, a process termed **threat hunting**.

Recent advances in Large Language Models (LLMs) have demonstrated impressive potential to augment cybersecurity practices, including malware analysis (Abusitta et al., 2021; Al-Karaki et al., 2024; Qian et al., 2025; Devadiga et al., 2023), penetration testing (Deng et al., 2023; 2024; Happe & Cito, 2023; Muzsai et al., 2024), and fuzzing (Zhang et al., 2025; Oliinyk et al., 2024; Black et al., 2024). Building on this progress, there is growing interest in 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 (Sehgal & Thymianis, 2023; Faghihi et al., 2023; Dash et al., 2022), 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 (MITRE Corporation,

Table 1: Comparison between CYBERTEAM and other LLM-oriented cybersecurity benchmarks.

Benchmark	Focus	#Data	#Task	#Source	Coverage	Unique Feature
CWE-Bench-Java (Li et al., 2025)	Java vulnerability	120	4	1	Four CWE classes	Large-scale Java codes
CTIBench (Alam et al., 2024)	Cyber Threat Intelligence	2,500	3	6	CVE, CWE, CVSS, ATT&CK	Multi-choice questions (MCQ)
SevenLLM-Bench (Ji et al., 2024)	Report understanding	91,401	28	N/A	Bilingual instruction corpus	Synthetic Data, MCQ, QA
SWE-Bench (Jimenez et al., 2023)	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, Standardized Reasoning Env

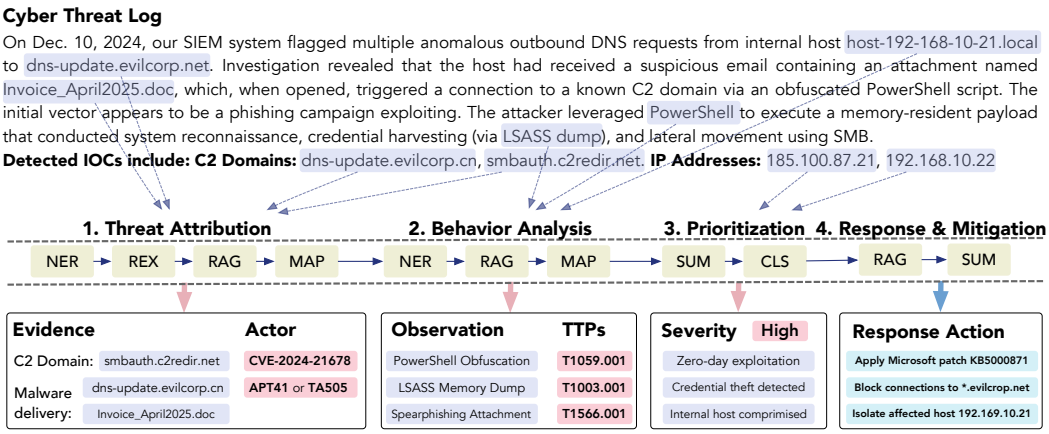

Figure 1: A CYBERTEAM threat hunting example equipped with operational modules. Module names: NER–named entity recognition, REX–regex parsing, MAP–text mapping, RAG–retrieval-augmented generation, CLS–classification, SUM–summarization.

2024), NVD (National Institute of Standards and Technology (NIST), 2024), and CISE (CISE Program, 2024), as well as security platforms such as Red Hat Bugzilla (Red Hat, Inc., 2024), Oracle Security Alerts (Oracle Corporation, 2024), and IBM X-Force (IBM Corporation, 2024). In addition, CYBERTEAM presents a larger number of tasks and samples than existing cybersecurity benchmarks (Jimenez et al., 2023; Li et al., 2025; Alam et al., 2024; Ji et al., 2024), 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.

Standardized Workflow. An important feature of CYBERTEAM is its structured, modular workflow for guiding LLMs within a standardized reasoning environment (Yang et al., 2024; Cheng et al., 2025). **This design is inspired by blue team practices, where analysts typically follow standardized procedures to interpret threat metadata and conduct investigations** (Sehgal & Thymianis, 2023; Diogenes & Ozkaya, 2018; Brotherston et al., 2024). However, strict adherence to such procedures can limit flexibility when analyzing unstructured threat logs or addressing emerging, zero-day threats. **To balance standardization and flexibility**, CYBERTEAM integrates a set of operational modules 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 operational modules. 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).

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 modularized approach with popular open-ended reasoning strategies such as In-Context Learning (ICL) (Dong et al., 2022), Chain-of-Thought (CoT) (Wei et al., 2022), Tree-of-Thought (ToT) (Yao et al., 2023). Our evaluation provides insights into the actionable threat hunting across 30 tasks.

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 a standardized reasoning workflow 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. To facilitate future research, we release codes at: `https://anonymous.4open.science/r/LLM-Cyberteam-7433/`.

2 Related Work

LLMs for Cybersecurity. Recently, LLMs have shown promise in enhancing cybersecurity tasks such as malware classification (Abusitta et al., 2021; Al-Karaki et al., 2024; Qian et al., 2025; Devadiga et al., 2023), code vulnerability detection (Russell et al., 2018; Lu et al., 2024; Sheng et al., 2024), penetration testing (Happe & Cito, 2023; Muzsai et al., 2024; Shen et al., 2024), phishing detection (Kulkarni et al., 2024; Greco et al., 2024), and incident report generation (Bernardi et al., 2024; Sufi, 2024; McGregor et al., 2025). 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 (Mündler et al., 2023; Simhi et al., 2025; Shrivastava), raising concerns about reliability in high-stakes defensive scenarios.

Cybersecurity Benchmarks. Recent benchmarks have focused on static analysis (Reinhold et al., 2024; Higuera et al., 2020; Braga et al., 2017), software vulnerabilities (Hossen et al., 2024; Sawant et al., 2024), and threat report generation (Tihanyi et al., 2024; Perrina et al., 2023; Čupka et al., 2023). These benchmarks evaluate predefined tasks such as identifying CWE categories, matching CVEs, or summarizing intelligence reports (Alam et al., 2024; Aghaei et al., 2020; Branescu et al., 2024; Hemberg et al., 2020). 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 (Fei et al., 2023; Wang et al., 2024; Choshen et al., 2024; Lucas et al., 2024; Zhou et al.). Inspired by these efforts, we introduce CyberTeam to emphasize structured reasoning and realistic interdependencies for blue teaming scenarios.

Operation-Guided Agents. Recent research has proposed agents with operational modules to structure LLM reasoning into modular, interpretable steps (Driess et al., 2023; Dongre et al., 2024; Hu et al., 2024). Such frameworks have achieved notable success in robotics (Jeong et al., 2024; Akkaladevi et al., 2021), database querying (Kadir et al., 2024; Dar et al., 2019), and scientific reasoning tasks (Abate et al., 2020; Vaesen & Houkes, 2021). However, their use in cybersecurity, especially defensive operations, remains underexplored despite the need for structured workflows. Our work addresses this gap by introducing a modular 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 modularized strategy (3.3).

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 (MITRE Corporation, 2024)). 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.

Table 2: Threat hunting tasks, description of targets, corresponding modularized operations, number of instances, and evaluation metrics. Details of implemented 9 modules and involved metrics are in Appendix B and C, respectively.

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

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.

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 (FIRST, a)) 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 (National Institute of Standards and Technology (NIST), 2024), MITRE CVE (The MITRE Corporation, n.d.), ATT&CK (MITRE Corporation, 2024), CWE (MITRE Corporation, b), CAPEC (MITRE Corporation, a), D3FEND (MITRE Corporation, c), Exploit-DB (Offensive Security, 2024), and VulDB (VulDB Team, 2024).

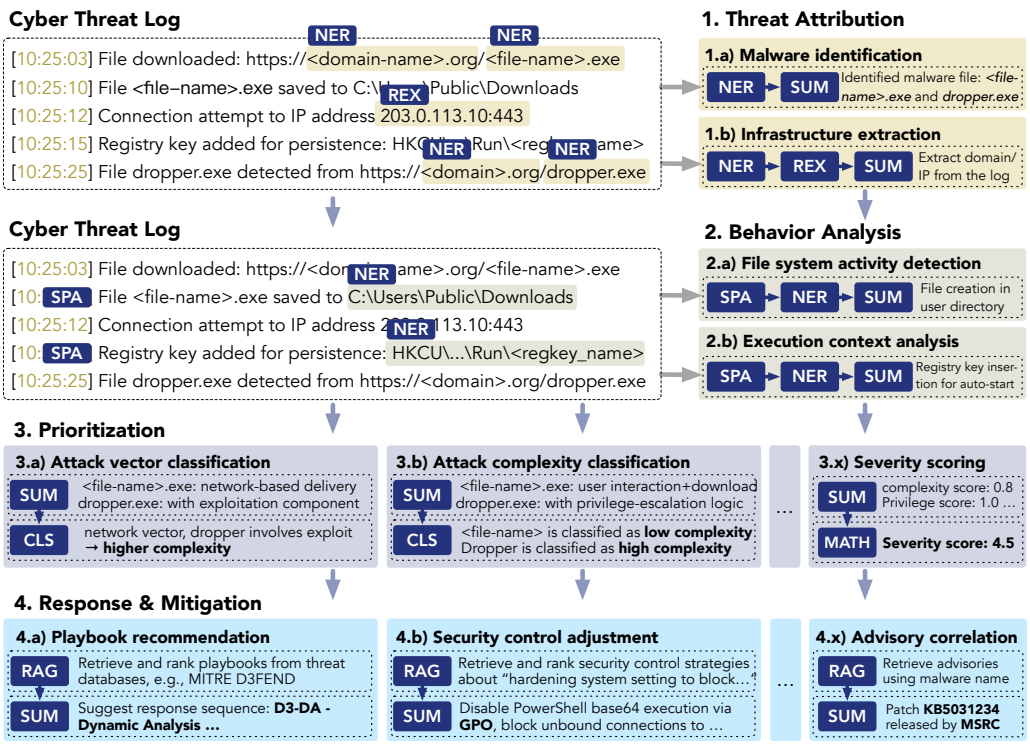

Figure 2: A threat hunting example demonstrating a dependency chain of analytical tasks, where each task is completed through a sequence of operational modules executed by LLMs autonomously.

These sources include detailed insights such as exploitability scores (EPSS (FIRST, b)), severity metrics (CVSS (FIRST, a)), and remediation guidance. Additionally, we incorporate data from vendor-maintained repositories (e.g., the Microsoft Security Update Guide (Microsoft Corporation), IBM X-Force (IBM Corporation, 2024)) 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 (VirusTotal (Google Chronicle), 2024), AlienVault OTX (AlienVault (AT&T Cybersecurity), 2024), and MISP (MISP Project, 2024) 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 (Mandiant (Google Cloud), 2024), Recorded Future (Recorded Future, 2024), Palo Alto Unit 42 (Palo Alto Networks, 2024), and Apache (The Apache Software Foundation, 2024), 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 Standardized Threat Hunting with Operational Modules

Task Dependency. Threat hunting is inherently a multi-stage analytical process (Sauerwein et al., 2019; Caltagirone et al., 2013; Hillier & Karroubi, 2022), 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.

Highlight 💡. Instead of enumerating all tasks listed in Table 2, LLMs are asked to determine which tasks to perform at each stage, opening reasoning flexibility in threat hunting.

Operational Modules. Within each threat hunting task, CYBERTEAM invokes a set of operations (functional modules) designed to produce actionable threat analyses and progressively address the current threat hunting target (e.g., incident response). Specifically, each threat hunting task t_i is associated with a corresponding set of operational modules $\mathcal{F}_i = \{f_i^1, f_i^2, \dots\}$. Each task t_i involves executing a 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.

Highlight 💡. These modules provide **broad coverage of threat hunting practices** (as shown in Table 2), while retaining flexibility (e.g., in SUM, RAG) for LLM reasoning to adapt across diverse scenarios, thereby **balancing flexibility with standardization** in blue team threat hunting.

Due to space constraints, we defer implementation details and design rationales to Appendix B.

4 EXPERIMENT

CYBERTEAM aims to empirically address the following research questions: **RQ$_1$:** How effective is standardization compared to open-ended reasoning for threat-hunting tasks? **RQ$_2$:** Can LLMs accurately solve individual threat-hunting tasks? **RQ$_3$:** 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 (G4o), GPT-o4-mini (Go4), Qwen3-32B (QW), Gemini-2.5 (GM), Claude-Sonnet-4 (CD), Llama-3.1-405B (L3.1), Llama-4-Scout-17B (L4), and Gemma-3-27b (GA). In addition, we assess state-of-the-art cybersecurity-focused LLM agents, including Lily-Cybersecurity-7B (LY) (Segolily Labs, 2025), DeepHat-7B (DH) (DeepHat, 2025), and SevenLLM-7B (SL) (Ji et al., 2024).

Open-ended Reasoning. In open-ended reasoning, we consider three widely used prompting structures: (1) In-Context Learning (ICL) (Dong et al., 2022) – including basic task instructions along with five (or ten) illustrative examples to demonstrate the desired solution format. (2) Chain-of-Thought (CoT) (Wei et al., 2022) – encouraging the model to generate "step-by-step" reasoning results before producing the final answer. (3) Tree-of-Thought (ToT) (Yao et al., 2023) – guiding LLMs to explore multiple reasoning paths and select the most plausible one.

Metrics. Table 2 lists evaluation metrics tailored to 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** (Zhang* et al., 2020) to measure 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., severity scoring) are evaluated using **Normalized Distance** to quantify similarity to ground truth values. All metrics are scaled to the range [0, 1]. We explain the rationale for those metrics in Appendix C.

4.1 STANDARDIZED THREAT HUNTING VS. OPEN-ENDED REASONING (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

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** to indicate the best results and underline to denote the second-best results.

Method		Cybersecurity Agent			Industry-Leading LLM							
		LY	DH	SL	G4o	Go4	QW	GM	CD	L3.1	L4	GA
Playbook Recommend												
Open-ended	ICL5	42.3$_{\pm 1.9}$	54.2$_{\pm 2.4}$	54.7$_{\pm 2.3}$	64.5$_{\pm 2.7}$	73.1$_{\pm 2.9}$	52.8$_{\pm 2.0}$	79.4$_{\pm 1.8}$	63.7$_{\pm 2.1}$	65.8$_{\pm 2.6}$	55.8$_{\pm 2.0}$	54.9$_{\pm 2.1}$
	ICL10	44.1$_{\pm 1.8}$	52.5$_{\pm 2.6}$	55.3$_{\pm 2.7}$	65.2$_{\pm 2.8}$	74.5$_{\pm 2.9}$	53.6$_{\pm 2.1}$	80.2$_{\pm 1.9}$	64.9$_{\pm 2.2}$	66.4$_{\pm 2.7}$	56.4$_{\pm 2.1}$	55.5$_{\pm 2.2}$
	CoT	51.6$_{\pm 2.2}$	50.6$_{\pm 2.4}$	50.5$_{\pm 2.2}$	78.3$_{\pm 1.9}$	89.2$_{\pm 1.8}$	67.5$_{\pm 2.5}$	80.1$_{\pm 1.6}$	81.4$_{\pm 1.7}$	77.3$_{\pm 2.0}$	67.3$_{\pm 2.3}$	66.4$_{\pm 2.2}$
	ToT	48.1$_{\pm 2.4}$	53.3$_{\pm 2.6}$	54.3$_{\pm 2.5}$	75.2$_{\pm 2.2}$	85.1$_{\pm 1.9}$	71.4$_{\pm 2.4}$	83.5$_{\pm 1.7}$	77.2$_{\pm 2.0}$	82.1$_{\pm 1.8}$	72.1$_{\pm 2.1}$	71.2$_{\pm 2.2}$
Standardized (Ours)		**67.2**$_{\pm 1.7}$	**58.4**$_{\pm 2.1}$	**66.8**$_{\pm 1.9}$	**84.6**$_{\pm 1.5}$	**91.4**$_{\pm 1.4}$	**79.3**$_{\pm 2.0}$	**91.8**$_{\pm 1.3}$	**89.3**$_{\pm 1.6}$	**89.7**$_{\pm 1.5}$	**79.7**$_{\pm 1.9}$	**78.8**$_{\pm 2.0}$
Security Control Adjust												
Open-ended	ICL5	51.5$_{\pm 2.4}$	66.3$_{\pm 2.3}$	43.9$_{\pm 2.7}$	61.8$_{\pm 2.6}$	70.3$_{\pm 0.7}$	50.6$_{\pm 2.2}$	65.8$_{\pm 2.1}$	79.2$_{\pm 1.9}$	61.5$_{\pm 2.4}$	51.5$_{\pm 2.0}$	50.6$_{\pm 1.1}$
	ICL10	53.2$_{\pm 2.5}$	68.4$_{\pm 2.4}$	45.6$_{\pm 2.8}$	62.7$_{\pm 2.0}$	71.8$_{\pm 2.8}$	51.2$_{\pm 0.1}$	66.4$_{\pm 2.2}$	80.1$_{\pm 1.8}$	62.3$_{\pm 2.4}$	52.3$_{\pm 2.1}$	51.4$_{\pm 2.1}$
	CoT	60.3$_{\pm 2.0}$	70.5$_{\pm 2.1}$	68.4$_{\pm 1.9}$	70.3$_{\pm 2.0}$	80.2$_{\pm 1.8}$	59.8$_{\pm 2.4}$	79.2$_{\pm 1.7}$	77.2$_{\pm 1.9}$	77.9$_{\pm 1.8}$	67.9$_{\pm 1.4}$	63.0$_{\pm 2.1}$
	ToT	66.7$_{\pm 1.9}$	72.1$_{\pm 2.0}$	61.6$_{\pm 2.2}$	75.9$_{\pm 1.8}$	85.6$_{\pm 1.0}$	66.3$_{\pm 2.3}$	73.6$_{\pm 2.1}$	73.1$_{\pm 2.0}$	72.8$_{\pm 0.3}$	62.8$_{\pm 2.3}$	61.9$_{\pm 2.2}$
Standardized (Ours)		**74.2**$_{\pm 1.6}$	**77.6**$_{\pm 1.5}$	**80.1**$_{\pm 1.7}$	**82.1**$_{\pm 1.6}$	**89.7**$_{\pm 1.2}$	**74.7**$_{\pm 1.8}$	**88.5**$_{\pm 1.5}$	**86.5**$_{\pm 0.7}$	**86.4**$_{\pm 1.7}$	**76.4**$_{\pm 1.8}$	**75.5**$_{\pm 0.9}$
Patch Code Generation												
Open-ended	ICL5	10.8$_{\pm 1.0}$	49.8$_{\pm 3.1}$	29.2$_{\pm 2.7}$	56.2$_{\pm 2.8}$	58.4$_{\pm 1.6}$	39.3$_{\pm 2.9}$	63.7$_{\pm 2.2}$	47.5$_{\pm 2.5}$	49.2$_{\pm 0.6}$	39.2$_{\pm 2.9}$	38.3$_{\pm 3.8}$
	ICL10	12.6$_{\pm 1.6}$	51.2$_{\pm 3.0}$	31.5$_{\pm 2.8}$	57.8$_{\pm 2.7}$	59.1$_{\pm 1.5}$	40.1$_{\pm 1.8}$	64.9$_{\pm 0.3}$	48.6$_{\pm 0.4}$	50.1$_{\pm 1.1}$	40.1$_{\pm 1.7}$	39.2$_{\pm 2.7}$
	CoT	24.5$_{\pm 2.2}$	54.7$_{\pm 2.4}$	55.1$_{\pm 2.1}$	58.4$_{\pm 2.4}$	76.3$_{\pm 1.9}$	54.7$_{\pm 2.3}$	65.3$_{\pm 2.0}$	66.3$_{\pm 2.0}$	67.4$_{\pm 1.8}$	57.4$_{\pm 2.1}$	51.5$_{\pm 2.2}$
	ToT	25.3$_{\pm 2.1}$	50.9$_{\pm 2.5}$	58.3$_{\pm 2.0}$	61.8$_{\pm 2.3}$	72.5$_{\pm 2.0}$	50.2$_{\pm 2.5}$	69.8$_{\pm 1.9}$	61.4$_{\pm 2.1}$	62.9$_{\pm 2.0}$	52.9$_{\pm 2.3}$	52.2$_{\pm 2.3}$
Standardized (Ours)		**29.7**$_{\pm 1.9}$	**63.4**$_{\pm 1.8}$	**60.2**$_{\pm 2.0}$	**72.5**$_{\pm 1.7}$	**87.4**$_{\pm 1.3}$	**65.4**$_{\pm 2.0}$	**82.6**$_{\pm 1.4}$	**79.2**$_{\pm 1.6}$	**80.6**$_{\pm 1.5}$	**70.6**$_{\pm 1.8}$	**69.7**$_{\pm 1.9}$
Patch Tool Suggestion												
Open-ended	ICL5	48.2$_{\pm 2.6}$	65.2$_{\pm 2.3}$	61.5$_{\pm 2.5}$	68.9$_{\pm 2.2}$	79.4$_{\pm 1.9}$	59.2$_{\pm 2.4}$	74.1$_{\pm 2.1}$	68.5$_{\pm 2.3}$	70.3$_{\pm 2.3}$	60.3$_{\pm 2.4}$	59.4$_{\pm 2.6}$
	ICL10	49.1$_{\pm 2.5}$	64.7$_{\pm 2.4}$	63.1$_{\pm 2.6}$	69.7$_{\pm 2.2}$	80.6$_{\pm 1.9}$	60.3$_{\pm 2.4}$	74.9$_{\pm 2.0}$	69.8$_{\pm 2.2}$	71.4$_{\pm 2.2}$	61.4$_{\pm 2.4}$	60.5$_{\pm 2.5}$
	CoT	53.6$_{\pm 2.2}$	70.1$_{\pm 2.1}$	77.2$_{\pm 1.8}$	79.2$_{\pm 1.7}$	90.1$_{\pm 1.5}$	70.3$_{\pm 2.0}$	81.7$_{\pm 1.6}$	79.1$_{\pm 1.8}$	79.6$_{\pm 1.9}$	69.6$_{\pm 2.0}$	68.7$_{\pm 2.1}$
	ToT	56.5$_{\pm 2.0}$	71.8$_{\pm 2.0}$	68.1$_{\pm 2.2}$	75.8$_{\pm 1.9}$	86.3$_{\pm 1.7}$	74.5$_{\pm 1.9}$	86.3$_{\pm 1.5}$	83.7$_{\pm 1.2}$	84.2$_{\pm 1.6}$	74.2$_{\pm 1.8}$	67.3$_{\pm 2.2}$
Standardized (Ours)		**69.1**$_{\pm 1.8}$	**76.5**$_{\pm 1.7}$	**77.7**$_{\pm 1.7}$	**87.4**$_{\pm 1.4}$	**96.9**$_{\pm 1.2}$	**83.6**$_{\pm 1.6}$	**93.2**$_{\pm 1.3}$	**91.2**$_{\pm 1.5}$	**92.1**$_{\pm 1.4}$	**82.1**$_{\pm 1.6}$	**81.2**$_{\pm 1.7}$
Advisory Correlation												
Open-ended	ICL5	21.7$_{\pm 3.8}$	57.5$_{\pm 2.6}$	63.8$_{\pm 2.5}$	64.7$_{\pm 2.4}$	67.2$_{\pm 2.4}$	48.5$_{\pm 3.3}$	62.4$_{\pm 2.6}$	56.8$_{\pm 2.7}$	58.7$_{\pm 2.6}$	48.7$_{\pm 3.1}$	47.8$_{\pm 3.2}$
	ICL10	22.9$_{\pm 3.7}$	59.1$_{\pm 2.6}$	64.7$_{\pm 2.5}$	65.9$_{\pm 2.4}$	68.1$_{\pm 2.3}$	49.2$_{\pm 3.1}$	63.2$_{\pm 2.5}$	58.1$_{\pm 2.6}$	59.5$_{\pm 1.2}$	49.5$_{\pm 3.1}$	48.6$_{\pm 3.2}$
	CoT	49.5$_{\pm 2.3}$	71.4$_{\pm 2.0}$	69.5$_{\pm 2.1}$	67.2$_{\pm 2.2}$	80.5$_{\pm 1.8}$	61.7$_{\pm 2.4}$	77.5$_{\pm 1.8}$	76.2$_{\pm 1.9}$	76.3$_{\pm 0.9}$	66.3$_{\pm 2.1}$	65.4$_{\pm 1.1}$
	ToT	46.8$_{\pm 2.3}$	73.2$_{\pm 1.9}$	67.2$_{\pm 2.2}$	70.8$_{\pm 2.0}$	84.2$_{\pm 1.7}$	64.8$_{\pm 2.2}$	73.1$_{\pm 1.9}$	72.5$_{\pm 2.0}$	71.8$_{\pm 2.1}$	61.8$_{\pm 2.3}$	60.9$_{\pm 2.3}$
Standardized (Ours)		**73.4**$_{\pm 1.7}$	**78.8**$_{\pm 1.5}$	**77.1**$_{\pm 1.6}$	**80.3**$_{\pm 1.5}$	**92.3**$_{\pm 1.3}$	**76.5**$_{\pm 1.8}$	**86.9**$_{\pm 1.4}$	**84.5**$_{\pm 1.6}$	**84.9**$_{\pm 1.5}$	**74.9**$_{\pm 1.7}$	**74.0**$_{\pm 1.7}$

mitigation outputs on CYBERTEAM. Table 3 presents the results, using task-specific metrics detailed in Table 2.

Effectiveness of Standardization. From Table 3, we observe that using operational modules (**Ours**) outperforms typical open-ended reasoning methods. For instance, modular operations enable GPT-o4 to achieve over 90% Hit@10 in playbook recommendation and over 92% in advisory correlation. In contrast, open-ended reasoning achieves only secondary effectiveness, with a significant performance gap observed (e.g., in the security control adjustment task of SevenLLM). This demonstrates the effectiveness of combining standardized guidance with the inherent flexibility of LLMs.

Gains from Standard Operating Procedures. Notably, while ICL, CoT, and ToT have been shown to improve generation quality for general-purpose tasks (Dong et al., 2022; Yu et al., 2023; Wang et al., 2022), they provide limited guidance for domain-specific problems that require precise procedural knowledge and structured analytical workflows. By contrast, standardized threat hunting workflows help LLMs follow standard operating procedures by decomposing complex tasks into modular steps. This reduces hallucination and enforces structure. In tasks requiring strict sequencing (e.g., threat actor identification followed by response planning), workflow-based methods ensure the correct order and information flow, outperforming ICL, CoT, and ToT, which often lack such control.

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, operations in CYBERTEAM constrain LLM reasoning to resolve correct analytical sequences, ensuring outputs remain aligned with operational goals:

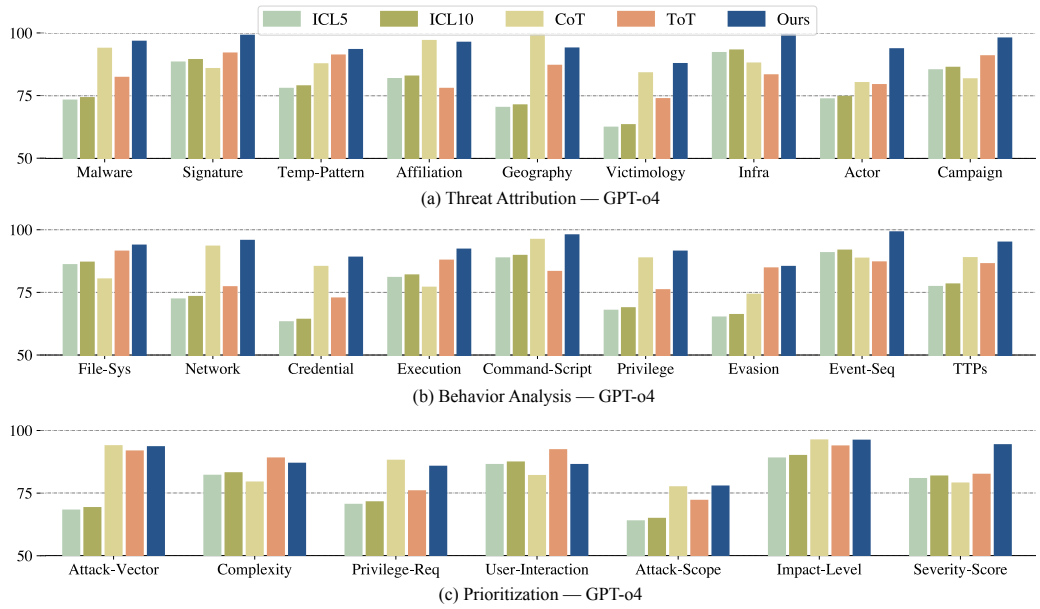

Figure 3: Threat-hunting performance (scaled to 100%) on individual tasks, evaluating under GPT-o4-mini. Results for additional LLMs are provided in Appendix E.

Case Study II (Successful Case). The modular operation 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.

Operational Interpretability. Notably, the modular approach enhances interpretability for analysts, as outputs can be traced back to specific operations (e.g., RAG for evidence retrieval, SUM for summarization). In contrast, open-ended prompts produce opaque reasoning chains that are harder to audit what real-world evidence is integrated.

Case Study III (Interpretability). For the MOVEit vulnerability (CVE-2023-34362), an open-ended Qwen prompt returned only a vague recommendation ("apply vendor patches and monitor suspicious traffic"). In contrast, our pipeline invoked the **RAG** module to retrieve Progress Software's advisory and the **NER** module to extract SQL injection IOCs. This modular trace improved accuracy and enabled analysts to audit advisory steps.

Due to space constraints, we provide additional evaluation of the trade-off between latency and reliability in Appendix E.1. Our results show that the standardized threat hunting method achieves a more favorable balance compared with open-ended reasoning.

Design Insights. The evaluation provides two actionable insights for blue team practices: (1) Breaking threat analysis into smaller, modularized operations (e.g., IOC extraction, TTP mapping), each guided by distinct reasoning objectives; (2) Integrate LLMs into existing analytic pipelines where upstream outputs (e.g., extracted indicators) are fed into downstream modules rather than relying on single-pass generation.

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 E present the experimental results.

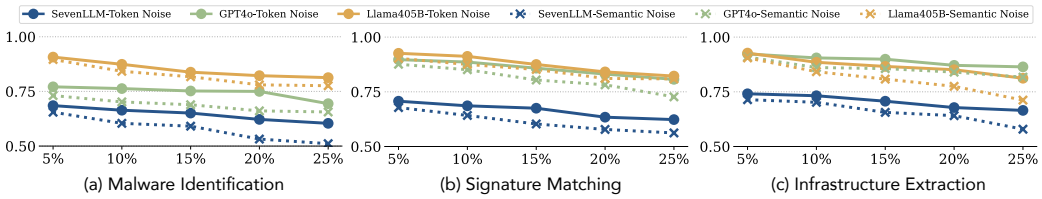

Figure 4: LLM performance (metrics corresponding to Table 2) when input threat logs are perturbed with token-level noise (solid line) or semantic-level noise (dashed line). X-axis shows the noise ratios.

Observe that using standardized threat hunting 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), the standardized method yields substantial improvements over open-ended reasoning strategies like CoT and ToT, boosting accuracy by over 20% using GPT-o4-mini. These gains are most notable when task dependencies are strong. For example, generating effective responses depends on accurate upstream analysis. Module-guided models can preserve and pass critical context, while ICL/CoT/ToT often fail to coordinate such multi-stage reasoning reliably. This is largely because these tasks require multi-hop reasoning, evidence synthesis, 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 operational modules 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.

Design Insights 💡**.** While standardized threat hunting offer general advantages, their relative benefit is particularly significant in **scenarios requiring structured reasoning** over interconnected steps. This demonstrates the importance of modular guidance in complex cybersecurity workflows.

Due to space constraints, we provide complementary results and analyses in Appendix E.2.

4.3 LLM Robustness against Noisy Inputs (RQ_3)

Experimental Setting. We also investigate LLM robustness when input threat logs contain noisy text. We introduce (i) token-level noise using TextAttack (Morris et al., 2020), which randomly injects or substitutes tokens, and (ii) semantic-level noise using BART-paraphraser (Lewis et al., 2019), 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 4, 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 (e.g., paraphrased or subtly altered context) causes a much larger decline. These findings suggest that while LLMs handle surface-level errors relatively well, they struggle with 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, a standardized workflow 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 offer actionable insights for integrating standardized operations into security workflows. We hope CyberTeam will serve as a valuable resource for the research community and practitioners alike, driving future innovations in AI-assisted cybersecurity.

Ethics Statement

This study is based solely on publicly accessible cybersecurity reports, vulnerability databases, and open-source intelligence platforms, each used in accordance with their copyright and licensing conditions. No proprietary, sensitive, or personally identifiable data were collected or processed.

Reproducibility Statement

To ensure reproducibility, we provide an anonymous GitHub repository containing benchmark construction details, operational module implementations, evaluation pipelines, and experiment configurations.

References

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.

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.

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

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.

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.

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.

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

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)*, pp. 1–8. IEEE, 2024.

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.

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)*, pp. 170–181. IEEE, 2017.

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

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

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

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.

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*, pp. 19–25, 2024.

CISE Program. Cybersecurity information sharing environment (cise), 2024. URL https://www.cisa.gov/cybersecurity-information-sharing.

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*, pp. 313–348. Springer, 2023.

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.

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.

DeepHat. Deephat-v1-7b. Model on Hugging Face, 2025.

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.

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)*, pp. 847–864, 2024.

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)*, pp. 53–58. IEEE, 2023.

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

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.

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.

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.

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, pp. 9552–9561, 2023.

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.

FIRST. Common vulnerability scoring system (cvss). https://www.first.org/cvss/, a.

FIRST. Exploit prediction scoring system (epss). https://www.first.org/epss/, b.

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.

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*, pp. 2082–2086, 2023.

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.

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.

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.

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

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.

IBM Corporation. Ibm x-force exchange, 2024. URL `https://exchange.xforce.ibmcloud.com/`.

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

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.

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.

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.

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.

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.

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.

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.

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.

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

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, pp. 28938–28945, 2025.

Microsoft Corporation. Microsoft security update guide. https://msrc.microsoft.com/update-guide.

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

MITRE Corporation. Common attack pattern enumeration and classification (capec). https://capec.mitre.org/, a.

MITRE Corporation. Common weakness enumeration (cwe). https://cwe.mitre.org/, b.

MITRE Corporation. D3fend: A knowledge graph of cybersecurity countermeasures. https://d3fend.mitre.org/, c.

MITRE Corporation. Mitre att&ck framework, 2024. URL https://attack.mitre.org/.

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*, pp. 119–126, 2020.

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.

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

National Institute of Standards and Technology (NIST). National vulnerability database (nvd), 2024. URL https://nvd.nist.gov/.

Offensive Security. Exploit database (exploit-db), 2024. URL https://www.exploit-db.com/.

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)*, pp. 883–900, 2024.

Oracle Corporation. Oracle security alerts, 2024. URL https://www.oracle.com/security-alerts/.

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

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)*, pp. 3053–3062. IEEE, 2023.

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.

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)*, pp. 285–288. IEEE, 2011.

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

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

Red Hat, Inc. Red hat bugzilla, 2024. URL https://bugzilla.redhat.com/.

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.

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)*, pp. 757–762. IEEE, 2018.

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.

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

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

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.

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.

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.

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

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.

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

The Apache Software Foundation. Apache security advisories. https://www.apache.org/security/, 2024.

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

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)*, pp. 296–302. IEEE, 2024.

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.

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

VulDB Team. Vuldb vulnerability database, 2024. URL https://vuldb.com/.

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.

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.

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.

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*, pp. 16227–16237, 2024.

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.

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.

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.

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.

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. URL https://arxiv.org/abs/1904.09675. Official implementation: https://github.com/Tiiiger/bert_score.

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

A.1 Data Source

The MITRE CVE (Common Vulnerabilities and Exposures) system (The MITRE Corporation, n.d.) 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)** (National Institute of Standards and Technology (NIST), 2024) 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 (Offensive Security, 2024) 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) (MITRE Corporation, b) 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. The metadata includes weakness category, severity, and relationships with CAPEC and CVE entries.

CAPEC (Common Attack Pattern Enumeration and Classification) (MITRE Corporation, a) 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 (MITRE Corporation, 2024) 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 (MITRE Corporation, c) 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 (Oracle Corporation, 2024) 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 (Red Hat, Inc., 2024) 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) (Red Hat, Inc.) 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 (IBM Corporation, 2024) 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) (CISE Program, 2024), 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) (VulDB Team, 2024) 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) (The Apache Software Foundation, 2024). 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 (Mandiant (Google Cloud), 2024), 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 (Recorded Future, 2024) 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) (Palo Alto Networks, 2024) 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 (Microsoft Corporation) 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) (FIRST, a) 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) (FIRST, b), 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) (MISP Project, 2024) 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 (VirusTotal (Google Chronicle), 2024) 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) (AlienVault (AT&T Cybersecurity), 2024) 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.

Data Ethics. All data used in CYBERTEAM are collected from publicly available vulnerability databases and open-source threat intelligence platforms. No sensitive personal information or proprietary organizational data are included.

A.2 DATA PREPARATION

Collection & Processing Settings. Our benchmark integrates threat intelligence from two primary categories of sources: (1) structured vulnerability databases and (2) semi-structured threat intelligence platforms. For the former, we collect machine-readable feeds from authoritative repositories such as NVD, MITRE CVE, CWE, CAPEC, D3FEND, Exploit-DB, VulDB, and EPSS. These sources provide canonical identifiers (e.g., CVE-ID), structured metadata (e.g., CVSS vectors, exploit maturity, CWE type), and standardized taxonomies that serve as grounding for downstream tasks. For the latter, we ingest reports, advisories, and campaign summaries from industry platforms including VirusTotal, AlienVault OTX, Mandiant, Unit 42, IBM X-Force, Cisco Talos, and Recorded Future. These sources contain higher-level context such as TTPs, indicators of compromise (IoCs), adversary attribution, and remediation actions. All raw sources undergo normalization (UTF-8 encoding, HTML stripping) before task-specific extraction, during which we decompose documents into evaluation units aligned with one of the five defined threat-hunting tasks. This pipeline ensures that the benchmark combines comprehensive structured knowledge with real-world narrative intelligence.

Validation & Quality Control. We apply a multi-stage validation pipeline to ensure the reliability and semantic consistency of all benchmark entries. Structured vulnerability feeds undergo schema validation to guarantee the completeness and format correctness of required fields (e.g., CVSS vectors must contain all eight base metrics; exploit maturity must match predefined categorical values). For threat reports, we apply heuristic validation steps such as cross-referencing CVE identifiers with NVD, enforcing ATT&CK technique format compliance (Txxxx), and verifying timestamp consistency. Additionally, we perform cross-source coherence checks to ensure that metadata reported by intelligence platforms matches the underlying vulnerability or campaign identifiers when available. To complement automated validation, human verification is applied to sampled entries from each source category to confirm contextual accuracy and to remove non-actionable or irrelevant content (e.g., marketing or promotional blog posts). This layered validation approach guarantees that all benchmark samples are both structurally valid and operationally meaningful.

Deduplication. Because multiple sources often report the same underlying security event, we implement deduplication at three levels. First, structural deduplication collapses entries that reference the same canonical identifier (e.g., CVE-ID, Exploit-DB ID, Malware hash) into a single base record that consolidates metadata across sources. Second, behavioral deduplication removes near-duplicate threat reports by computing MinHash signatures and merging samples whose Jaccard similarity exceeds 0.85, ensuring that mirrored advisories or lightly reworded vendor posts do not inflate dataset size or bias evaluations. Third, we apply task-instance deduplication after report decomposition: if a single report contributes to multiple task categories, each instance is retained exactly once per task, and the final data statistics reflect unique evaluation units rather than repeated documents. This deduplication strategy prevents redundancy while preserving the richness of multi-perspective threat intelligence.

B MODULARIZED OPERATIONS: BASIC COMPONENT OF STANDARDIZED THREAT HUNTING

To support modular and extensible capabilities within our CYBERTEAM, we decompose complex NLP workflows into discrete, modularized operations. This section detail the implementation of NLP modules as described in section 3.1. Each module 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, specifically:

Implementation. At execution time, the agent calls NER through inputs containing the raw threat events (e.g., log snippet, CTI fragment), a threat-hunting stage identifier (e.g., attribution, behavior analysis, response planning), or a configurable entity schema. The LLM is then instructed to return a structured JSON-compatible object that adheres to this schema. To enforce reliability, the function is wrapped with automatic retry logic and schema validation; if the model returns malformed fields or missing keys, the system invokes a repair cycle using a lightweight self-correction prompt. This keeps the module robust to hallucinations while avoiding the cost of supervised fine-tuning.

Prompt Template: NER for Cyber Threat Hunting

System Prompt:
You are a cybersecurity threat-hunting assistant with expertise in extracting structured security intelligence from noisy or unstructured text. Your task is to identify and categorize all cybersecurity-relevant entities present in the input and return them in a machine-readable format.
Instructions: Given the input text, extract all relevant named entities and classify them into the following categories (include only those that appear):

- **Threat Actor:** Names or labels of attacker groups or individuals (e.g., APT28, TA505, Lazarus).
- **Malware / Tool:** Malware families, implants, exploits, payloads, living-off-the-land tools (e.g., Cobalt Strike, Mimikatz).
- **Vulnerability:** CVE identifiers, zero-days, protocol flaws.
- **Infrastructure / IOC:** IPs, domains, URLs, file hashes, registry keys, cloud resources, container images.
- **Technique / TTP:** MITRE ATT&CK IDs, exploitation patterns, privilege escalation, lateral movement.
- **Target / Asset:** Affected platforms, operating systems, cloud accounts, business units.
- ...<Exhaustive list omitted>...

Output Format:
Return a JSON object with the following structure:

```
{
  "threat_actor": [...],
  "malware_or_tool": [...],
  "vulnerability": [...],
  "infrastructure": [...],
  "technique_or_ttp": [...],
  "target_or_asset": [...],
  ...<Exhaustive list omitted>...
}
```

Each value must be a list. Use empty lists if no entities are found.
Important Notes:

- Do not infer or hallucinate entities not grounded in the text.
- Normalize formats (e.g., lowercase domains, uppercase CVEs).
- Remove duplicates and irrelevant entities.
- Do not include explanations or reasoning in the output.

User Input:
<INSERT TEXT HERE>

Design Rationale: In real-world threat hunting, analysts are constantly overwhelmed by unstructured reports, logs, and advisories filled with technical jargon and entity references. Automating named entity recognition helps blue teams immediately isolate critical items such as threat actors, malware strains, or CVE identifiers without combing through entire reports manually. This reduces analyst workload, accelerates attribution, and ensures no important entity slips through, particularly when adversaries recycle or slightly modify names and indicators across campaigns.

B.2 REX (Regex Parsing)

The Regex Parsing (REX) module is responsible for extracting syntactically well-defined threat indicators from unstructured logs, reports, emails, or telemetry feeds using a library of predefined regular expressions. Unlike semantic NER, which delegates pattern understanding to a language model, REX enforces high-precision, rule-based extraction for entities that follow strict syntactic formats such as IPv4/IPv6 literals, domain names, URLs, file hashes (MD5/SHA1/SHA256), registry paths, email addresses, timestamps, MITRE ATT&CK IDs, and CVE identifiers. This makes REX particularly well suited for early-stage pipeline normalization, IOC pivoting, forensic processing, and cross-document correlation.

Implementation. The REX module is deployed as a lightweight function-wrapped operator inside the agent runtime. When invoked, the LLM is not responsible for pattern recognition itself; instead, the model triggers a function containing a curated regex registry indexed by indicator type. These patterns are compiled once at initialization using Python's re module or Rust-based regex engines for performance. The input text is streamed through a parsing pipeline, where each regex class is applied either sequentially or in parallel depending on configured performance constraints. Matches are normalized, deduplicated, and stored in a structured return schema that downstream modules (e.g., RAG or MAP) can consume.

Prompt Template: Regex-Based Indicator Extraction

System Prompt:
You are a cybersecurity log parsing assistant. Your task is to extract structured threat indicators using deterministic pattern matching (regex). Do not infer or guess values—only return data that matches well-formed patterns.
Instructions: Given the text below, extract all instances matching the following indicator classes:

- **IP Address:** IPv4 or IPv6 literals
- **Domain / URL:** Fully qualified domain names or URLs
- **File Hash:** MD5, SHA1, SHA256
- **CVE Identifier:** Format CVE-YYYY-NNNNN
- **Timestamp:** Any ISO, syslog, Windows, or common log format
- **ATT&CK Technique:** Format TXXXX[.XXX]
- ...<Exhaustive list omitted>...

Output Format:
Return a JSON object in the following schema:

```
{
  "ip": [...],
  "domain": [...],
  "url": [...],
  "file_hash": [...],
  "cve": [...],
  "timestamp": [...],
  "attack_technique": [...],
  ...<Exhaustive list omitted>...
}
```

Rules:

- Output only syntactically valid matches.
- Do not include any reasoning or natural language explanation.
- Normalize formats (e.g. lowercase domains, uppercase CVEs, uppercase hashes).
- Return empty lists for categories with no matches.
- Deduplicate all results.

User Input:
<INSERT TEXT HERE>

Fault Tolerance. The REX function supports runtime extension, enabling operators to dynamically inject new patterns or adjust matching rules without retraining any models. In cases where extraction ambiguity exists (e.g., whether a string is a file hash or random hex), the system performs confidence checks via post-processing heuristics or performs an optional validation pass through the LLM for schema correction. Because REX outputs structured artifacts rather than free-form text, it also provides reliable anchor points for dependency tracking and pipeline auditing, enabling consistent behavior across threat hunting tasks from attribution to mitigation.

Design Rationale: Regex parsing remains indispensable because many threat indicators—such as IP addresses, hashes, and domains—follow strict syntactic patterns. Blue team analysts often must quickly normalize raw log data or incident feeds into structured formats suitable for correlation across SIEM or TIP platforms. Automated regex-based extraction delivers high precision and avoids false alarms, providing reliable building blocks for pivoting investigations, linking disparate alerts, and enriching threat databases with verified observables.

B.3 SUM (SUMMARIZATION)

The Summarization (SUM) module focuses on compressing lengthy, often repetitive cybersecurity narratives into analyst-ready intelligence while preserving operationally critical content. In contrast to general-purpose summarization, SUM is optimized for blue-team workflows and retains domain-specific elements such as TTP identifiers, IOCs, victims, exploit chains, and mitigation references.

The SUM function is also task-aware: depending on the stage in the dependency chain (e.g., behavior analysis vs. response & mitigation), it adjusts its compression ratio and focus. Summaries are stored and propagated forward, e.g., the MAP function uses prior summaries to map indicators to security tools or ATT&CK techniques. Because the module is prompt-based rather than model-specific, it can be upgraded or extended without retraining, and it also supports multi-format summarization (single paragraph, bullet points, timeline, executive brief).

Implementation. SUM operates as a controlled LLM inference wrapped in a function call it's applied via a summarization prompt with explicit coverage requirements.

Prompt Template: Structured Cyber Threat Summarization

System Prompt:
You are a cybersecurity threat intelligence analyst. Your task is to distill the essential technical and investigative details from the following report while preserving operational relevance.

The summary will be fed into downstream analysis modules, so factual precision and coverage are critical.
Instructions: Summarize the report into a concise paragraph (3–5 sentences) that includes:

- Attack vector(s) used (e.g., phishing, RCE, supply chain)
- Key TTPs, tools, payloads, or malware families
- Indicators of compromise (IPs, domains, hashes, filenames)
- Affected systems, platforms, or business units
- Timeline elements (dates, sequence of notable events)
- Threat actor or attribution context (if mentioned)
- ...<Exhaustive list omitted>...

Output Format:

- Output a single plain-text summary paragraph.
- Do not include reasoning steps or formatting.
- Do not add information not grounded in the text.
- Avoid generic cybersecurity language.
- ...<Exhaustive list omitted>...

User Input:
<INSERT REPORT TEXT HERE>

Fault Tolerance. We further check whether the final output includes required elements like TTPs or IOCs; if not, a corrective round is triggered automatically using a refinement prompt. This enables analysts to obtain high-value summaries even when reports contain redundant text, marketing language, or narrative filler.

Design Rationale: Threat reports and advisories are typically lengthy, verbose, and include redundant or irrelevant details. In time-critical investigations, analysts need condensed yet accurate snapshots that retain attack vectors, key actors, and affected assets. Automated summarization provides blue teams with quick situational awareness, enabling them to brief stakeholders or prioritize triage without missing essential context. It also helps align tactical actions with strategic threat intelligence by stripping away noise and surfacing the essentials.

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.

Implementation. SIM is invoked as a lightweight operation that accepts text pairs and returns a structured similarity judgment. While LLMs perform the reasoning, the module enforces a constrained output schema (boolean + confidence + justification) to support downstream automation. Confidence calibration is handled through a scoring strategy—models may either (1) generate a self-assessed confidence score or (2) be wrapped with a probabilistic calibration layer (e.g., logit normalizers or temperature scaling) to produce stable outputs for downstream classifiers. In chain-of-thought restricted environments, the justification can be generated independently using controlled prompting.

Prompt Template: Threat Intelligence Similarity Matching

System Prompt:
You are a cybersecurity threat intelligence analyst. Your task is to determine whether two textual descriptions refer to the same underlying entity, region, or concept in a cyber threat context. Focus on meaning, not word overlap.
Instructions: Given two text snippets, decide whether they semantically refer to the same threat origin or classification. Consider the following criteria:

- Do both terms refer to the same region, culture, language group, or geopolitical sphere?
- Would threat intelligence analysts commonly treat them as interchangeable or equivalent?
- Does one term logically imply the other (e.g., hierarchical or subset relationship)?
- ...<Exhaustive list omitted>...

Output Format (JSON):

```
{
  "match": true/false,
  "confidence": <float from 0.0 to 1.0>,
  "justification": "<one or two sentences>"

}
```

Rules:

- Do not hallucinate geopolitical facts not grounded in common CTI usage.
- Avoid vague language; be explicit and concise.
- If uncertain, return "false" with lower confidence.
- ...<Exhaustive list omitted>...

User Input: "text-a": "<PHRASE 1>", "text-b": "<PHRASE 2>"

Design Rationale: Threat hunting often suffers from inconsistent terminology—analysts and vendors may describe the same adversary group or region in different ways. By applying semantic similarity matching, blue teams can unify aliases, regional descriptions, or contextual cues, thereby avoiding fragmented analysis. For example, detecting that "Eastern European actors" and "Russian-speaking threat groups" likely refer to the same set of adversaries allows more coherent attribution and prevents intelligence silos that adversaries can exploit.

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.

Implementation. MAP is implemented as a controlled function call that prompts an LLM to convert threat report text into relational triples using a constrained grammar. To ensure consistency, MAP operates on top of the structured output of upstream modules (e.g., NER, REX) and inherits their normalization. When reports are long, the system chunk-summarizes relevant sections before triple extraction. Triples are validated using schema rules and optionally cross-referenced with public ontologies (e.g., MITRE ATT&CK, CAPEC, CVE metadata) to filter impossible or low-confidence edges.

Prompt Template: Knowledge Graph Triple Extraction for Threat Intelligence

System Prompt:
You are a cybersecurity knowledge mapping assistant. Your task is to convert the threat report into structured triples suitable for knowledge graph construction. Focus on relationships that matter for attribution, behavior analysis, and mitigation planning.
Instructions: From the text below, extract subject–predicate–object triples, using only information grounded in the text. Examples of valid predicates include (but are not limited to):

- **uses / deploys / distributes** (actor → malware/tool)
- **exploits / targets** (tool or actor → vulnerability or victim)
- **communicates with / hosts / controls** (tool → infrastructure or C2)
- **linked to / attributed to / associated with** (malware or campaign → threat actor)
- **observed in / active since** (indicator → date or region)

Output Format (JSON Array):

```
[
  {
    "subject": "...",
    "predicate": "...",
    "object": "...",
    "confidence": <float between 0.0 and 1.0>
  },
  ...
]
```

Rules:

- Triples must be grounded in the text — no speculation or external facts.
- Keep entities normalized (e.g., uppercase CVEs, lowercase domains).
- Avoid vague predicates like "related to" if a more precise one applies.
- Use a new entry for each distinct triple — do not chain multiple relations into one object.
- If uncertain, include the triple with a low confidence value rather than omitting it.

User Input: <INSERT REPORT TEXT HERE>

Design Rationale: Attack campaigns rarely consist of isolated events—they are orchestrated through complex infrastructures and actor-tool relationships. Mapping extracted entities into structured knowledge graphs helps analysts visualize these relationships and trace adversary activity across time and geography. This capability supports detection of infrastructure reuse, identification of campaign evolution, and discovery of hidden connections that might otherwise remain unnoticed, enabling more proactive defense strategies and long-term threat tracking.

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.

Implementation. The RAG module is implemented as a hybrid retrieval system that combines a local, index-backed threat intelligence store with real-time access to external APIs.

Case 1: To support fast and semantically relevant lookup, we construct a vector database using FAISS as the primary indexing backend. Incoming documents including CVE entries, MITRE ATT&CK profiles, malware analyses from threat intelligence vendors, advisories from CISA, and campaign reports from industry sources (exhaustive list in Appendix A) are first normalized and split into passages. Each passage is embedded using a domain-optimized sentence transformer, and both

embeddings and metadata (source, timestamp, threat type, confidence flags) are stored for retrieval. A lightweight SQLite or Elasticsearch layer stores metadata alongside embeddings, enabling structured filtering based on source authority, document recency, and entity type. This indexing strategy allows RAG to efficiently retrieve highly relevant evidence even when dealing with hundreds of thousands of threat intelligence records.

Case 2: Beyond local indexing, the module supports external retrieval for evolving threats that may not yet exist in the offline database. To do this, the LLM generates a structured search query, which is passed to external providers such as VirusTotal, OTX, Shodan, Recorded Future, or even search engines scoped with domain restrictions (e.g., site:mitre.org or site:cisa.gov). Retrieved results are normalized and converted into candidate passages, which are embedded and re-ranked against internal results using a hybrid scoring function combining dense similarity, BM25 keyword matching, and optional LLM-based relevance scoring. If external sources fail to respond or return low-quality results, the system automatically falls back to the local vector index, maintaining robustness and availability during inference.

The final generation process is executed as a two-step pipeline. First, the system returns a retrieval context composed of top-ranked passages and metadata. Then, the LLM is prompted to synthesize an answer grounded strictly in retrieved evidence. To prevent hallucination, citations are enforced through schema validation, and responses are rejected if they contain assertions unsupported by retrieval. The module also logs all retrieved evidence into the shared working memory layer, enabling downstream modules such as MAP for threat mapping or SUM for report summarization to reuse the same evidence without redundant retrieval. By combining local indexing with external live search and pairing retrieval with generation under strict grounding constraints, the RAG module supports high-fidelity, up-to-date analysis while preserving transparency and auditability in blue-team threat hunting.

Design Rationale: Adversary tactics evolve daily, and static LLMs quickly become outdated if disconnected from real-time sources. Retrieval-augmented generation enables blue teams to ground LLM outputs with fresh, authoritative information from trusted CTI feeds, vulnerability databases, or public repositories. This ensures that generated insights remain both accurate and timely, supporting decisions during live incidents such as phishing outbreaks or zero-day exploitation campaigns where stale intelligence could lead to ineffective responses.

B.7 SPA (Text Span Localization)

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

Implementation. SPA first uses semantic search to narrow down text windows containing relevant context, then applies a prompt-based extraction pass where the LLM returns the minimal span satisfying the query. For improved precision, the extracted span is validated using token-based alignment heuristics that measure similarity against known technique glossaries (e.g., MITRE ATT&CK descriptions). If the output is too vague, over-extended, or incomplete, a refinement loop is triggered to reissue the prompt with tightened constraints. Outputs are scored using two complementary metrics: Exact Match (EM) for strict correctness, and Intersection-over-Union (IoU) to measure partial overlap between predicted and ground-truth spans. Detailed formulations are provided below:

- **Exact Match (EM)**:

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

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

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

Prompt Template: Targeted Span Extraction for Cyber Threat Reports

System Prompt:
You are a cybersecurity span localization assistant. Your goal is to extract the minimal text

span that directly describes the attack technique used in the incident. The output must match the source text exactly, not a paraphrase.
Evaluation Criteria (Built Into the Task): Your extracted span will be evaluated using two metrics:

- **Exact Match (EM):** You get full credit only if your extracted text exactly matches the ground truth span with no extra or missing tokens.
- **Intersection over Union (IoU):** Partial credit is awarded based on the overlap between your extracted span S_p and the true span S_t:

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

To maximize both scores, return the shortest possible span that fully captures the technique description without adding unrelated text.
Instructions: From the text below, extract the exact sentence or phrase that describes the primary attack technique or method of compromise (e.g., spearphishing, RCE, credential dumping, lateral movement). The output must be a direct substring of the input.
Output Format:

`"<EXACT_SPAN_FROM_TEXT>"`

Rules:

- Return only the span — no commentary or explanation.
- The extracted text must be contiguous and appear exactly in the input.
- If multiple spans qualify, return the most complete or specific one.
- If no valid span is present, return an empty string.

User Input: <INSERT REPORT EXCERPT HERE>

Design Rationale: In practice, analysts often need to pull out the single critical phrase—such as the exact exploitation method—from long reports or alerts. Span localization ensures precision by targeting actionable fragments rather than broad summaries, which is vital for creating detection rules, YARA signatures, or SIEM correlation logic. By pinpointing exact techniques or IOCs, blue teams reduce ambiguity, streamline evidence curation, and avoid wasting resources on imprecise or overly generalized intelligence.

B.8 CLS (CLASSIFICATION)

To measure the ability of a system to categorize cybersecurity-relevant textual inputs into predefined classes (e.g., threat categories, severity levels, or attack types), classification models are employed.

Implementation. We implement CLS 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.

Design Rationale: Blue teams constantly receive heterogeneous data ranging from phishing alerts to vulnerability disclosures. Automated classification allows this information to be triaged into relevant categories (e.g., attack type, severity, or impacted systems) so that workflows can be routed efficiently. Accurate classification supports prioritization of critical alerts, ensures compliance with response playbooks, and minimizes analyst fatigue by filtering out low-severity noise while surfacing the incidents that require immediate attention.

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.

Implementation. Considering that MATH is mainly used for severity quantification, we thus implement it through **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.

Design Rationale: Quantitative scoring frameworks like CVSS remain the backbone of enterprise vulnerability management and patch prioritization. Automated mathematical reasoning allows blue teams to consistently compute, validate, and apply these scores across large vulnerability sets, ensuring consistent triage even under heavy load. Beyond CVSS, mathematical modules enable probability modeling, risk scoring, and exposure forecasting—practices that help defenders allocate resources effectively and justify decisions to leadership with evidence-based metrics.

C METRIC

Below are further details on how each evaluation metric quantifies the corresponding threat hunting performance.

C.1 GENERATION (PRECISION–RECALL BALANCE BY F1) AND CLASSIFICATION (ACCURACY)

In threat hunting, information extraction tasks such as detecting malware names, extracting IOCs, or identifying exploited vulnerabilities require a careful balance between precision and recall. If a system retrieves too many irrelevant indicators, analysts are burdened with noise; if it misses critical signals, adversarial activity may go unnoticed. The **F1 score** captures this balance by evaluating how well a model retrieves the right items while minimizing both false alarms and missed detections. This makes it particularly valuable in operational contexts where the completeness and reliability of extracted intelligence directly affect the quality of subsequent analysis and response.

Besides, well-quantified tasks such as prioritization in blue team activities involve classification, such as determining whether an alert corresponds to privilege escalation, categorizing attack vectors, or assigning severity levels to vulnerability reports. In these scenarios, **accuracy** serves as an intuitive and effective measure of system performance, reflecting how often predictions align with ground-truth categories. High accuracy ensures that automated classification supports efficient triage and aligns with established taxonomies like MITRE ATT&CK.

C.2 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** (Zhang et al., 2020), 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.3 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.4 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.5 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 EXPERIMENTAL SETTING

This section details the experimental setup used to evaluate LLMs in the CyberTeam benchmark.

Hyperparameters. Table 4 summarizes the key hyperparameters for querying LLMs during experiments. These settings were chosen to balance generation quality and computational efficiency.

Table 4: LLM query hyperparameters.

Hyperparameter	Value	Description
Temperature	0.7	Output randomness
Top-p	0.95	Nucleus sampling threshold
Max tokens	2048	Generation length cap
Stop sequences	`["\n", "Q:"]`	Response cutoff cues
Prompt format	ICL, CoT, ToT, Emb	Prompt types (see 4)
Tool-calling API	Enabled (Selective)	For function-use experiments

Computational Resources. All experiments were conducted on a high-performance computing cluster equipped with six NVIDIA RTX 6000 Ada Generation GPUs, each with 48 GB of dedicated VRAM. The system utilized CUDA version 12.8 and NVIDIA driver version 570.124.06. This configuration enabled parallel execution of model inference, evaluation, and tool-augmented tasks across the benchmark datasets. The hardware provided sufficient memory bandwidth and processing power to handle large-scale experiments, including multi-sample prompting strategies like CoT and ToT, without encountering resource constraints. Each experimental run was executed in a isolated environment to ensure reproducibility and avoid interference between tasks.

E 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.

Table 5: Running time (in seconds) of LLMs on CYBERTEAM, comparing different open-ended prompting strategies with our standardized method. Lower values indicate faster inference.

Method		Cybersecurity Agent			Industry-Leading LLM							
		LY	DH	SL	G4o	Go4	QW	GM	CD	L3.1	L4	GA
Playbook Recommend												
Open-ended	ICL5	12.4	15.6	14.8	10.5	41.2	13.6	11.9	12.7	28.6	24.0	16.8
	ICL10	14.1	17.2	16.3	12.0	45.7	15.2	13.5	14.4	31.4	26.5	18.3
	CoT	18.6	22.4	21.1	15.8	60.8	19.9	17.6	18.8	41.2	34.5	24.5
	ToT	27.5	34.1	32.0	24.2	89.5	30.2	27.1	28.9	62.7	52.5	37.8
Standardized (Ours)		21.3	26.5	25.2	19.1	71.3	23.5	21.0	22.3	50.5	42.0	30.1
Security Control Adjust												
Open-ended	ICL5	13.1	16.4	15.5	11.1	43.5	14.3	12.5	13.3	30.2	25.0	17.6
	ICL10	15.0	18.3	17.2	12.6	48.1	16.1	14.2	15.1	33.0	27.4	19.1
	CoT	19.4	23.5	22.2	16.7	63.4	20.8	18.3	19.6	43.8	36.0	25.7
	ToT	28.3	35.6	33.6	25.4	92.2	31.7	28.3	30.2	66.4	55.0	39.5
Standardized (Ours)		22.1	27.8	26.7	20.0	74.2	24.7	22.1	23.4	53.1	44.0	31.2
Patch Code Generation												
Open-ended	ICL5	14.2	17.9	17.0	12.2	46.8	15.7	13.6	14.4	32.8	27.0	19.2
	ICL10	16.3	19.6	18.7	13.7	51.3	17.5	15.3	16.2	35.7	29.5	20.8
	CoT	21.2	25.3	24.5	18.1	68.7	22.9	19.7	21.1	47.6	39.0	28.1
	ToT	31.7	38.4	36.9	27.8	98.6	34.5	30.6	32.6	71.9	59.0	42.5
Standardized (Ours)		24.0	29.7	28.6	21.4	79.4	26.6	23.6	25.0	57.2	47.0	34.4
Patch Tool Suggestion												
Open-ended	ICL5	12.8	15.9	15.2	10.9	42.6	14.0	12.3	13.0	29.5	24.3	17.0
	ICL10	14.7	17.7	16.9	12.4	47.0	15.8	14.0	14.8	32.4	26.2	18.6
	CoT	19.0	23.0	22.0	16.3	62.1	20.5	18.1	19.2	42.5	35.0	25.1
	ToT	27.9	34.9	33.1	24.7	90.8	31.0	27.6	29.6	64.0	52.5	38.2
Standardized (Ours)		21.7	27.1	26.2	19.6	72.8	24.1	21.7	22.9	51.7	42.5	30.5
Advisory Correlation												
Open-ended	ICL5	13.6	16.8	16.1	11.7	44.9	14.9	13.0	13.8	31.0	25.5	18.2
	ICL10	15.6	18.7	17.9	13.2	49.6	16.7	14.7	15.6	34.0	28.0	20.0
	CoT	20.3	24.1	23.4	17.2	65.3	21.7	19.0	20.4	45.3	37.0	26.5
	ToT	29.8	36.8	35.4	26.1	95.1	33.1	29.4	31.3	68.8	56.0	40.1
Standardized (Ours)		23.1	28.5	27.8	20.6	76.2	25.4	22.8	24.2	54.6	45.0	32.1

E.1 RUNNING TIME AND TRADE-OFF BETWEEN LATENCY AND EFFECTIVENESS

Observations and Insights. The runtime analysis highlights an inherent trade-off between efficiency and reasoning complexity across prompting strategies. Consistent with expectations, in-context learning (ICL) variants remain the fastest across nearly all models, typically completing tasks in the 10–15 second range. This makes ICL attractive for time-sensitive operations such as triage or initial correlation, where speed outweighs the need for more structured reasoning. Chain-of-thought (CoT) introduces additional reasoning overhead, increasing runtimes by roughly 30–40% compared to ICL. While this slowdown is measurable, the benefit of CoT lies in its improved consistency on more nuanced decision tasks, suggesting that blue teams might selectively invoke CoT when precision is critical. Tree-of-thought (ToT), by contrast, incurs the highest latency, often doubling the runtime relative to ICL. This stems from ToT's multi-branch exploration process, which, while occasionally producing richer reasoning chains, remains computationally expensive and operationally impractical for most real-time security workflows.

Our standardized pipeline approach falls between CoT and ToT in runtime. The added latency reflects the sequential decomposition of tasks into modular subroutines, each enforcing more structured reasoning than raw prompting. While slower than single-pass approaches, **our pipeline mostly avoids the extreme overhead observed in ToT.** This stability is particularly important in operational settings: analysts can predictably plan around a known latency budget while still benefiting from higher reliability and repeatability of results.

Unlike open-ended reasoning, which may fluctuate in quality depending on the model and prompt, the standardized pipeline enforces uniform logic steps, reducing error propagation at the cost of additional inference time. From a deployment standpoint, this balance offers a pragmatic middle

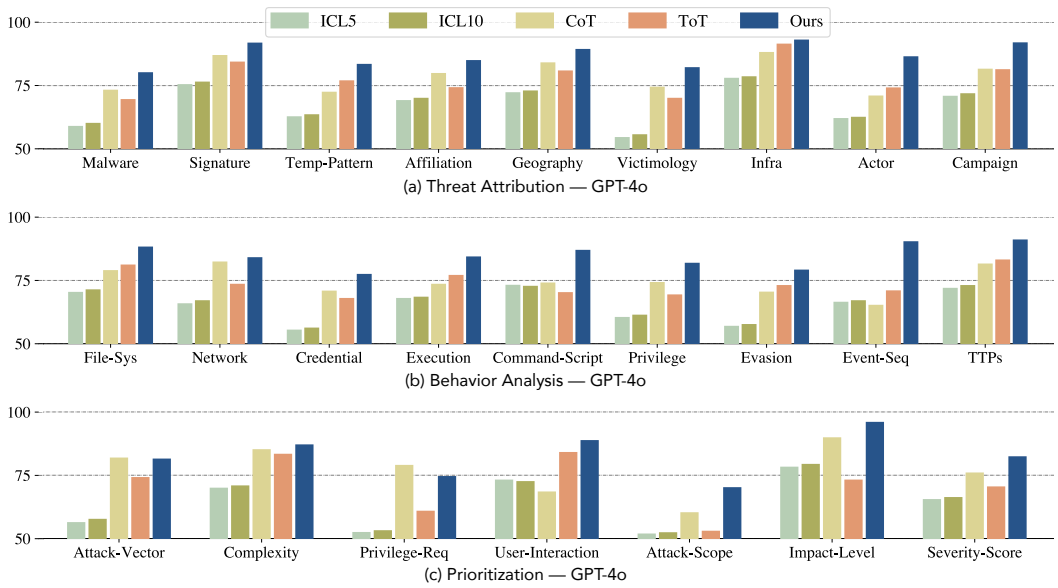

Figure 5: Threat-hunting performance on individual tasks, evaluating under GPT-4o.

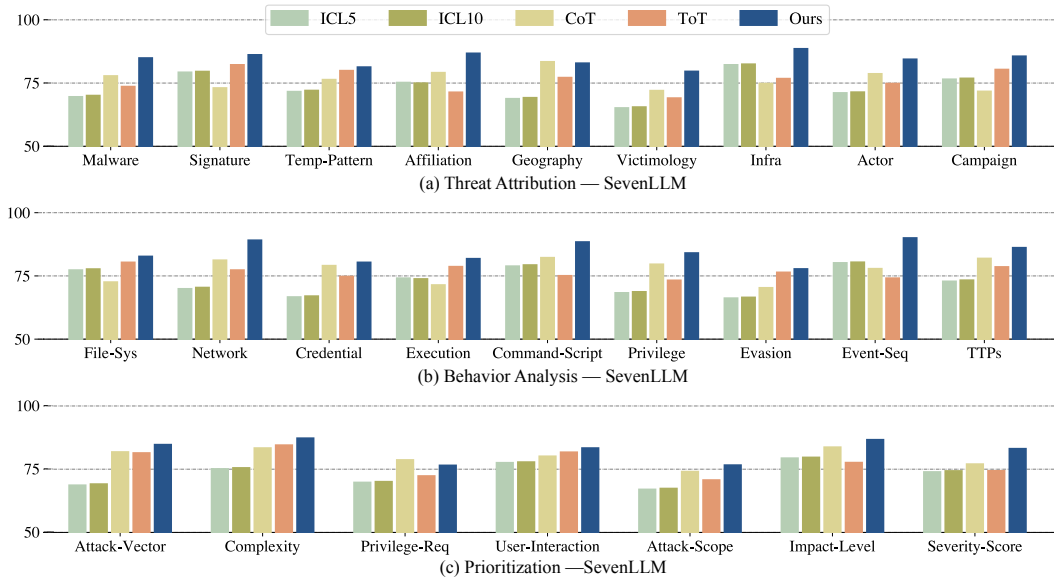

Figure 6: Threat-hunting performance on individual tasks, evaluating under SevenLLM-7B.

ground: not as lightweight as ICL for quick heuristics, but substantially more usable than ToT when analysts demand repeatable outputs.

E.2 Additional Results of Individual Threat Hunting Performance

Figure 5, 6, 7, and 8 complement the results as present in Figure 3, offering aligned insights as exhibited in previous experiments.

Based on those results, we further outline the following observations and analyses:

Attribution-Oriented Tasks. Attribution tasks rely on aligning disparate indicators into coherent profiles of adversaries, infrastructure, and campaigns. Here, the standardized workflow shows its

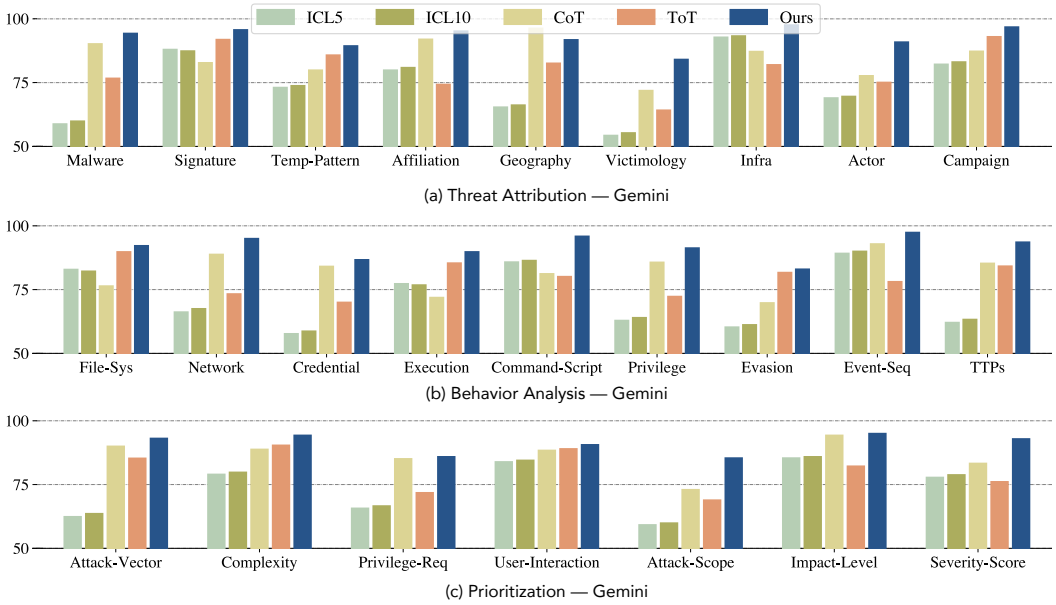

Figure 7: Threat-hunting performance on individual tasks, evaluating under Gemini-pro.

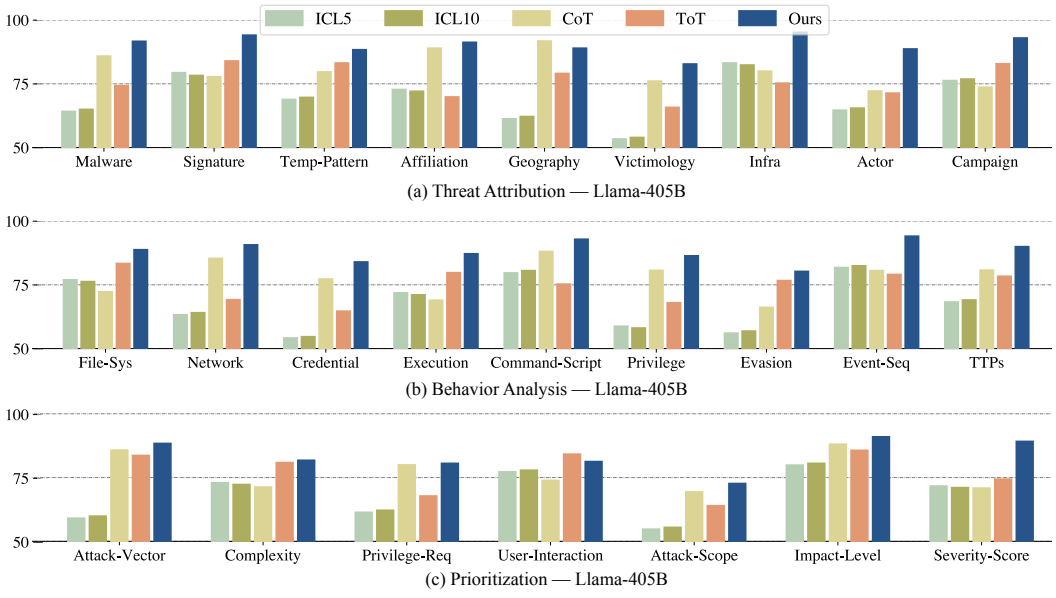

Figure 8: Threat-hunting performance on individual tasks, evaluating under Llama-405B.

greatest benefit because it forces the model to treat each extracted clue as part of a larger dependency chain. When the reasoning is left open ended, models often generate fluent narratives that omit critical ties, such as overlooking how infrastructure relates to a specific campaign or how victimology patterns reinforce an actor hypothesis. The modular approach ensures that entity recognition, context mapping, and relational inference are explicitly sequenced, which reduces the tendency of the model to drift or collapse multiple actors into a generic label. This structured pipeline also helps the model preserve continuity across steps, so that information about geography, malware signatures, or campaign overlaps is not forgotten or misapplied. In practice, this makes attribution outputs more consistent and trustworthy, with fewer contradictions across different facets of the same incident.

Behavioral-Oriented Tasks. Behavioral analysis tasks focus on describing how an attack unfolds across file systems, networks, credentials, execution flows, and evasion strategies. These tasks expose a different challenge: models must reason not just about isolated labels but about temporal or causal sequences. Open ended reasoning often struggles to maintain logical order, for example by misplacing the relationship between a credential theft and subsequent privilege escalation, or by skipping intermediate steps in an event sequence. Standardized workflows address this by explicitly guiding the model to construct event chains step by step, preserving both order and dependency. This guidance is particularly important when behaviors are nested, such as when command script execution spawns further lateral movements or when evasion strategies are intertwined with persistence mechanisms. The modular design ensures that contextual cues are not discarded midway, producing outputs that resemble the structured analysis human analysts expect. The gains here are not simply about accuracy but about interpretability, as the resulting narratives make it easier to understand how behaviors connect to form a complete attack path.

Prioritization-Oriented Tasks. Prioritization tasks require models to map extracted observations into judgments of impact, severity, or scope. These are less about narrative flow and more about logical consistency and rule following. While open ended reasoning can handle straightforward labels like user interaction requirements or attack complexity, it often falters when multiple inputs must be integrated into a composite assessment. For example, determining severity requires careful alignment of impact level, attack vector, and privilege requirements, which is difficult to achieve reliably without structured steps. The standardized workflow enforces this alignment by ensuring that each component assessment is produced systematically and then fed into the final prioritization judgment. As a result, the model is less likely to generate inconsistent or contradictory scores. The benefits are particularly visible in tasks that resemble rule based calculations or scoring rubrics, where the modular structure mirrors the procedural way that human analysts reason about risk.

Broader Implications. Viewed across these categories, a clear pattern emerges. Attribution tasks benefit most from the preservation of contextual dependencies across different indicators. Behavioral tasks gain from the ability to model temporal and causal structure. Prioritization tasks see improvements in logical consistency and integration of multiple criteria. Standardization does not change the core language modeling capability of these systems, but it channels their generative power into workflows that mirror how analysts actually think about security problems. This alignment between workflow design and task demands is the primary driver of the gains observed, and it demonstrates why modular guidance is most valuable when reasoning requires structured coordination across multiple dimensions.

Benchmark Generalizability. Although CyberTeam integrates data from 23 diverse threat intelligence sources, the benchmark is inherently constrained by the selected datasets and threat scenarios. For example, it may not fully represent emerging attack vectors, such as AI-powered phishing or supply chain compromises. Expanding the benchmark to include more recent and varied threat data, as well as cross-domain applications (e.g., IoT or cloud security), would enhance its utility for evaluating LLM generalization in broader cybersecurity contexts.

E.3 Human Evaluation

In human evaluation, we recruited three security analysts to assess LLM-generated outputs across four stages of the threat-hunting workflow: Threat Attribution, Behavior Analysis, Prioritization, and Response & Mitigation. Evaluators are asked to make binary accept/reject decisions to reflect real-world triage settings, and we report acceptance rates with inter-annotator variance to capture the uncertainty.

Table 6: Human evaluation results on the four threat-hunting categories (scaled to 100%). Each cell reports mean acceptance rate $\pm$ standard deviation across 3 human evaluators.

Task / Subtask	Go4	QW	GM	L4	LY
Threat Attribution					
Malware Identification	$82.1_{\pm 0.9}$	$78.5_{\pm 2.7}$	$72.6_{\pm 1.3}$	$60.3_{\pm 2.4}$	$52.9_{\pm 11.4}$
Infrastructure Extraction	$92.4_{\pm 0.5}$	$87.1_{\pm 4.9}$	$90.8_{\pm 3.3}$	$84.6_{\pm 2.1}$	$65.4_{\pm 3.1}$
Actor Identification	$90.2_{\pm 1.7}$	$76.8_{\pm 3.1}$	$93.9_{\pm 1.2}$	$78.4_{\pm 2.3}$	$75.8_{\pm 7.2}$
Campaign Correlation	$83.5_{\pm 6.2}$	$74.2_{\pm 3.4}$	$89.6_{\pm 1.6}$	$75.1_{\pm 2.9}$	$61.3_{\pm 4.0}$
Behavior Analysis					
File System Activity Detection	$81.3_{\pm 7.2}$	$75.4_{\pm 2.8}$	$89.6_{\pm 1.7}$	$77.8_{\pm 2.2}$	$73.5_{\pm 3.1}$
Credential Access Detection	$75.7_{\pm 6.3}$	$73.4_{\pm 4.3}$	$63.4_{\pm 2.8}$	$67.6_{\pm 3.3}$	$59.3_{\pm 4.5}$
Command & Script Analysis	$82.2_{\pm 2.2}$	$71.9_{\pm 3.4}$	$84.6_{\pm 2.1}$	$72.4_{\pm 3.5}$	$40.9_{\pm 3.9}$
Event Sequence Reconstruction	$73.6_{\pm 2.6}$	$65.1_{\pm 3.7}$	$73.2_{\pm 2.4}$	$70.3_{\pm 3.8}$	$53.2_{\pm 10.2}$
Prioritization					
Attack Complexity Summarization	$89.2_{\pm 1.7}$	$67.3_{\pm 3.3}$	$84.1_{\pm 2.2}$	$71.1_{\pm 2.9}$	$63.8_{\pm 4.1}$
Privileges Requirement Detection	$91.7_{\pm 4.9}$	$72.6_{\pm 3.2}$	$87.4_{\pm 1.5}$	$74.1_{\pm 3.0}$	$70.8_{\pm 3.6}$
Attack Scope Detection	$86.4_{\pm 2.1}$	$64.9_{\pm 3.8}$	$81.2_{\pm 2.4}$	$68.4_{\pm 3.1}$	$61.9_{\pm 4.0}$
Severity Scoring	$87.8_{\pm 1.9}$	$66.0_{\pm 3.6}$	$82.9_{\pm 2.5}$	$69.7_{\pm 3.4}$	$62.5_{\pm 4.3}$
Response & Mitigation					
Playbook Recommendation	$93.4_{\pm 0.8}$	$78.1_{\pm 2.9}$	$88.4_{\pm 1.5}$	$76.3_{\pm 2.6}$	$79.2_{\pm 3.0}$
Patch Tool Suggestion	$84.6_{\pm 2.8}$	$61.2_{\pm 4.2}$	$78.5_{\pm 2.6}$	$66.9_{\pm 3.6}$	$71.4_{\pm 4.7}$
Advisory Correlation	$91.9_{\pm 1.1}$	$77.8_{\pm 2.7}$	$87.6_{\pm 1.4}$	$74.2_{\pm 3.1}$	$77.5_{\pm 3.3}$

The results show that stronger general-purpose LLMs consistently achieve higher acceptance rates, particularly when guided by the standardized reasoning workflow, while smaller or domain-specialized models exhibit greater variance and reduced reliability. Tasks that simply require structured reasoning or multi-step synthesis (e.g., campaign correlation, event reconstruction, and severity scoring) result in higher disagreement among evaluators, whereas modular tasks like IOC extraction or playbook recommendation benefit more from standardized workflows. Overall, the findings support our claim that CYBERTEAM 's modular framework improves interpretability and operational alignment, but also reveal that human oversight remains essential, especially for complex threat hunting scenarios.

E.4 ABLATION STUDY

To understand the contribution of each system component, we conduct an ablation study by removing different critical function modules that are mostly intensively used: NER, SPA, SUM, or RAG. Each ablation keeps all other components unchanged and the same dataset and evaluation pipeline are used.

Across all tasks, removing RAG or SUM causes the most severe performance degradation, implying their necessity for evidence grounding and context compression. SPA remains important for localizing objectives in reasoning but impacts less than retrieval-driven modules. Removing NER shows moderate performance loss, which is more pronounced in tasks requiring IOC- or actor-level normalization (e.g., Playbook Recommendation, Patch Tool Suggestion), but less in abstract reasoning tasks. These results demonstrate that named entity normalization is beneficial but not as critical as retrieval or summarization, suggesting LLM capabilities in implicit-entity reasoning or lightweight NER alternatives.

E.5 TOKEN CONSUMPTION

Overall, the token consumption analysis in Table 8 highlights several important efficiency trade-offs across prompting strategies and model families, wherein Tree-of-Thought (ToT) may incur higher token usage due to its iterative branching and self-evaluation steps, often doubling the cost of standard Chain-of-Thought (CoT). CYBERTEAM sits between CoT and ToT for most tasks, or perform colsely as ToT. Notably, CYBERTEAM typically reduces generation tokens relative to ToT while achieving comparable or improved reasoning quality (as shown in Table 3, Figure 5, etc.).

Table 7: Ablation study results of LLM threat-hunting performance (scaled to 100%).

Task	Ablation	Go4	QW	GM	L4	LY
Playbook Recommend	w/o SPA	88.9	76.5	89.7	77.2	65.1
	w/o SUM	70.3	58.7	71.4	62.9	44.2
	w/o RAG	61.9	53.1	68.9	57.8	46.0
	w/o NER	73.4	61.2	75.8	66.3	51.7
Security Control Adjust	w/o SPA	87.1	72.9	86.3	73.5	71.4
	w/o SUM	68.5	60.3	72.8	64.1	57.5
	w/o RAG	71.8	63.9	75.2	67.3	54.9
	w/o NER	76.3	65.1	78.4	68.2	59.3
Patch Code Generation	w/o SPA	84.6	62.8	80.4	68.9	26.8
	w/o SUM	53.5	45.2	57.9	48.0	19.1
	w/o RAG	58.0	48.9	62.3	52.5	20.7
	w/o NER	62.7	50.3	66.1	54.8	23.9
Patch Tool Suggestion	w/o SPA	94.2	81.0	91.1	80.3	66.7
	w/o SUM	71.4	60.8	72.2	61.9	48.3
	w/o RAG	77.6	67.3	79.4	68.1	52.7
	w/o NER	82.5	70.6	85.2	72.7	55.1
Advisory Correlation	w/o SPA	89.6	74.1	84.4	72.1	71.0
	w/o SUM	75.2	62.3	70.5	60.2	50.8
	w/o RAG	69.4	58.5	66.3	54.7	49.2
	w/o NER	72.1	60.4	69.7	58.6	52.1

Table 8: Token consumption (generation tokens) across LLMs in CYBERTEAM for different prompting strategies. Lower values indicate lower inference cost.

Method	Cybersecurity Agent			Industry-Leading LLM							
	LY	DH	SL	G4o	Go4	QW	GM	CD	L3.1	L4	GA
Playbook Recommend											
CoT	1,218	1,561	1,492	2,387	2,926	2,658	2,703	2,515	3,203	3,371	1,994
ToT	2,935	3,412	3,298	5,721	6,174	5,932	6,018	5,712	6,307	5,189	3,031
Ours	2,181	2,476	3,205	6,091	5,568	5,283	5,301	6,069	5,582	4,801	2,298
Security Control Adjust											
CoT	1,271	1,598	1,532	2,411	2,968	2,744	2,811	2,597	3,293	3,441	2,954
ToT	2,901	3,412	3,289	5,683	6,107	5,932	6,028	5,713	6,321	6,718	4,012
Ours	2,121	2,443	2,361	4,016	5,661	6,248	5,303	4,072	6,269	4,892	4,513
Patch Code Generation											
CoT	2,834	3,231	2,169	3,671	4,148	3,882	3,951	3,772	4,563	4,832	4,287
ToT	5,643	5,384	5,793	8,347	9,198	8,673	8,912	8,573	9,654	10,276	9,441
Ours	6,607	6,068	5,971	8,821	8,412	7,064	9,189	9,921	7,642	7,983	7,316
Patch Tool Suggestion											
CoT	1,183	1,497	1,438	2,371	2,911	2,683	2,735	2,481	3,136	3,274	2,967
ToT	2,674	3,212	3,048	5,104	5,711	5,463	5,622	5,217	5,932	6,118	5,591
Ours	2,215	2,487	2,431	3,911	4,309	4,129	4,181	3,927	4,532	4,718	4,243
Advisory Correlation											
CoT	1,331	1,674	1,598	2,703	3,161	2,912	2,989	2,754	3,488	3,622	2,178
ToT	2,823	3,477	3,399	5,888	6,438	5,587	6,341	5,998	6,482	6,841	3,197
Ours	2,464	2,793	2,702	4,367	4,853	6,172	4,619	4,389	6,207	6,242	4,716

The efficiency gains of CYBERTEAM are more pronounced on smaller cybersecurity-tuned models than on frontier models (e.g., GPT-4o, Gemini, Claude, Llama-4), suggesting that structured prompting provides greater marginal benefit for resource-constrained deployments. However, for code-intensive tasks like Patch Code Generation, all methods lead to higher token usage, reflecting the inherent complexity of structured code reasoning and the additional context required for precise patch synthesis. These results suggest that our prompting strategy provides a practical middle ground between minimal prompting (CoT) and computationally expensive deliberation (ToT), which implies scalable deployment while preserving reasoning effectiveness.

E.6 EVALUATION ON NON-LLM APPROACHES

Table 9: Comparison of non-LLM baselines and CYBERTEAM across multiple benchmark tasks. Non-LLM methods includes TF-IDF for Malware Identification, Regex-based IOC extractor for Signature Matching, cosine similarity for Temporal Pattern Matching, and LSTM-based text classifiers for Classification tasks. CYBERTEAM uses LLM-based reasoning with modular workflows. Best performance per task is bolded.

Task	Non-LLM	CYBERTEAM (LLM)	CYBERTEAM (Full)
Malware Identification (F1 ↑)	11.8	86.2	**94.1**
Signature Matching (F1 ↑)	54.6	92.3	**99.2**
Temporal Pattern Matching (Sim ↑)	52.1	77.8	**93.4**
Attack Vector Classification (Acc ↑)	65.2	90.1	**94.5**
Attack Complexity Classification (Acc ↑)	48.2	81.4	**88.2**
Privileges Requirement Detection (Acc ↑)	36.9	75.7	**86.5**

To provide a meaningful non-LLM comparison, we evaluate each task against a classical baseline representative of existing automation used in security operations as detailed in Table 9.

Across all evaluated tasks, CYBERTEAM consistently outperforms the non-LLM baselines, often by a substantial margin. While classical methods perform reasonably well on narrowly scoped, pattern-based tasks such as signature matching, they struggle to generalize to semantically complex or context-dependent tasks such as malware attribution and CVSS component inference. In contrast, CYBERTEAM 's modular LLM-driven workflow yields strong gains in both accuracy and robustness by leveraging contextual reasoning and task-specific prompting. These results suggest that while traditional techniques remain useful for deterministic subtasks, LLM-based workflows provide significantly greater utility in full-spectrum threat analysis and operational decision-making.

F DISCUSSION

Generalization to Unseen and Adversarial Threats. While CYBERTEAM focuses on benchmarking LLM-assisted analysis of real-world threat-hunting data, we recognize that defenders must also handle zero-day attacks and adversarially perturbed intelligence. Our benchmark partially reflects this scenario by including emerging CVEs and attack patterns that do not appear in prompt demonstrations, enabling us to evaluate zero-shot generalization. Moreover, CYBERTEAM 's standardized operational workflow offers a generalizable process that applies to both known and novel threats. However, adversarially constructed inputs designed to systematically mislead investigation (e.g., IOC poisoning, semantic obfuscation) require a formal threat model and security evaluation framework beyond the scope of conventional benchmarking. We therefore highlight adversarial generalization as an open research challenge and an opportunity for future work at the intersection of AI security and cyber defense.

Failure Analysis. We conduct an additional failure-oriented analysis on two representative modules: (1) NER-based entity extraction and (2) RAG-based retrieval. In NER-based modules (e.g., Malware Identification, Infrastructure Extraction), misses often occur when threat entities appear in uncommon formats, nested constructs, or vendor-specific aliases (e.g., ransomware payload names embedded within file paths). These errors propagate downstream, degrading actor attribution and mitigation planning. RAG-based modules exhibit a different failure signature: when logs contain incomplete or ambiguous indicators, the retriever occasionally surfaces high-overlap but semantically irrelevant sources, leading to outdated or mismatched remediation suggestions despite strong generative reasoning. These observations suggest that improving domain-specific synonym handling and retrieval precision is as crucial as scaling LLM reasoning capabilities. Incorporating structured failure annotation into future iterations of CYBERTEAM will enable more robust diagnosis and principled defenses.

G LARGE LANGUAGE MODEL (LLM) USAGE DISCLOSURE

LLMs were employed exclusively for light grammar refinement and phrasing adjustments while preparing the manuscript. They were not involved in conceptual development, benchmark design, experiment execution, or result interpretation. All scientific ideas, methodological designs, and analyses were carried out independently by the authors. LLM usage was limited to minor textual polishing.