UNCOVERING FAILURE MODES OF LLM-ASSISTED CYBER THREAT INTELLIGENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are increasingly used to help security analysts manage the surge of cyber threats, automating tasks from vulnerability assessment to incident response. Yet in CTI workflows, reliability gaps remain substantial. Existing explanations often point to generic model issues (e.g., hallucination), but we argue the dominant bottleneck is the threat landscape itself: CTI is heterogeneous, volatile, and fragmented. Under these conditions, evidence is intertwined, crowdsourced, and temporally unstable, which are properties that standard LLM-based studies rarely capture. This paper presents a comprehensive empirical study of LLM failure modes in CTI reasoning. We introduce a heterogeneous categorization framework that robustly labels failure modes across the CTI lifecycle, avoiding the brittleness of automated LLM-as-a-judge pipelines. We identify three domain-specific cognitive failures: spurious correlations from superficial metadata, contradictory knowledge from conflicting sources, and constrained generalization to emerging threats. We validate these mechanisms via causal interventions and show that targeted defenses reduce failure rates significantly. Together, these results offer a concrete roadmap for building resilient, domain-aware CTI agents.

1 INTRODUCTION

We are living in an era of rapid digital transformation, where technological advances are accompanied by a growing prevalence of cyber threats. In recent years, the cyber threat landscape has shifted dramatically, with reported Common Vulnerabilities and Exposures (CVEs) increasing by an average of 25% annually (CVE Program, 2024). In 2025 alone, more than 48,000 vulnerabilities were reported. This surge is driven by the increasing complexity of IT systems (Stalnaker et al., 2024), widespread adoption of open-source software (Dam & Neumaier, 2023), and accelerating development cycles (Trinkenreich et al., 2025). Together, these trends expand the attack surface while making sole reliance on human analysts for vulnerability assessment and remediation increasingly infeasible.

Large language models (LLMs) have recently shown strong potential across a broad range of cyber threat intelligence (CTI) tasks. Through instruction fine-tuning and prompt-based automation, researchers have applied LLMs to threat analysis and decision-making (Zhang et al., 2023), code vulnerability detection (Du et al., 2024), and defense against network intrusions (Lavi et al., 2024). Despite these advances, substantial performance gaps remain across existing evaluations (Deng et al., 2024; Clairoux-Trepanier et al., 2024; Ji et al., 2024; Alam et al., 2024; Liu et al., 2025), suggesting that these limitations cannot be fully addressed through conventional model adaptation or prompt engineering alone. This raises a fundamental research question: **What intrinsic failure modes constrain the effectiveness of LLMs in supporting CTI tasks?**

Although failures such as hallucination (Ji et al., 2023; Martino et al., 2023; Friel & Sanyal, 2023) and jailbreaking (Niu et al., 2024; Coalson et al., 2024) stem from general deficiencies in LLMs and training methodologies that affect many domains (Aguilera-Martínez & Berzal, 2025), studying these limitations in a domain-agnostic manner does not yield effective cyber-specific solutions. Developing LLM-based CTI agents requires reasoning over a threat landscape characterized by heterogeneous and volatile knowledge sources (Rigaki et al., 2024; De Santis et al., 2025). Unlike static corpora, CTI forms a fragmented ecosystem of structured databases (e.g., NVD (National Institute of Standards and Technology (NIST), 2024)), semi-structured vendor advisories (Berndtsson, 2017), and unstructured discussions on social media and dark web forums (Wang & Chow, 2019; Rahman et al., 2020).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

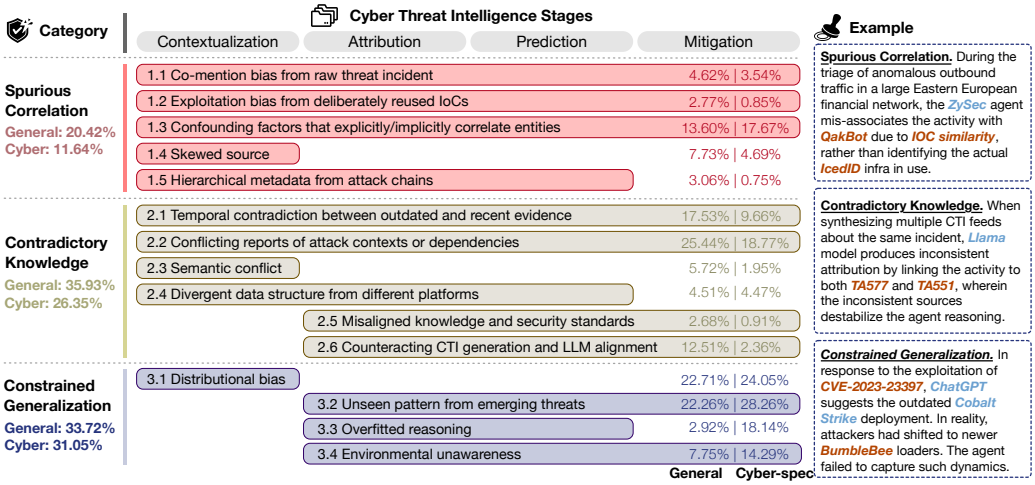


Figure 1: Summary and examples of LLM failure mode categories across different CTI stages, wherein we compare general-purpose LLMs and cyber-specific models. Ratios are calculated over the entire dataset (§2.2). **Several failures may occur simultaneously in a single threat instance.**

Moreover, cyber threats evolve continuously: the severity and remediation status of a vulnerability can change rapidly as new exploits emerge (Pureti, 2022). Consequently, standard LLM evaluations often fail to capture the disparate, time-sensitive, and frequently contradictory nature of the threat landscape.

This work. Motivated by the need for CTI agents to reason over intertwined, crowdsourced, and imbalanced evidence, we study: (1) which failure modes uniquely arise from the threat landscape itself; (2) how these factors induce specific weaknesses in LLM-assisted CTI; and (3) which defensive strategies can effectively mitigate them. We make the following contributions:

I. A comprehensive empirical evaluation. To characterize LLM failure modes in CTI, we conduct a large-scale, systematic study spanning the full CTI lifecycle, from threat analysis to incident response. Our evaluation combines established benchmarks: CTIBench (Alam et al., 2024), SevenLLM-Bench (Ji et al., 2024), SWE-Bench (Jimenez et al., 2023), and CyberTeam (Liu et al., 2025) with real-world threat reports from authoritative knowledge bases and threat feeds (e.g., (CVE Program, 2024; National Institute of Standards and Technology (NIST), 2024; Cybersecurity and Infrastructure Security Agency (CISA), 2024)), providing broad and realistic coverage.

II. A failure categorization methodology. Building on our evaluation results, we develop a rigorous methodology for identifying and categorizing failure instances. A key challenge in analyzing large-scale threat data is reliably scaling the classification of diverse failure modes. Standard LLM-as-a-judge approaches are unreliable for this task, as models may rationalize their own outputs or fail to critically assess contradictions (Yamauchi et al., 2025; Guerdan et al., 2025). To address this limitation, we propose an autoregressive framework that categorizes failure instances efficiently and reliably (§3). This methodology enables data-grounded analysis at scale while substantially reducing manual annotation effort.

III. A taxonomy of CTI failure modes and root cause analysis. Through large-scale experiments on industry-leading LLMs and LLM-powered cybersecurity agents, complemented by detailed case studies, we identify three dominant categories of failure modes that limit LLM reasoning: **(1) Spurious correlations**, where LLMs over-attribute based on superficial or co-occurring features (e.g., treating commodity tools as actor-specific evidence); **(2) Contradictory knowledge**, where inconsistencies across CTI sources lead to unstable or conflicting predictions; and **(3) Constrained generalization**, where LLMs struggle to extrapolate beyond familiar distributions, particularly on emerging or zero-day attack surfaces.

Our analysis further shows that these failure modes directly undermine core techniques used throughout CTI pipelines. As summarized in Figure 1, each failure mode spans multiple CTI stages and affects distinct aspects of LLM reasoning. For example, spurious correlations distort evidence re-

Example: CTI stages

A healthcare network detects suspicious outbound traffic linked to newly registered domains associated with *QakBot*. **The security team first enriches event metadata** by mapping proxy logs to known *C2 infrastructure* and retrieving prior reports for context. **Next, they attribute** *QakBot*'s reuse of infrastructure to the threat group *TA577*. Based on historical correlations, **they then predict** a likely transition to *Cobalt Strike* and eventual ransomware deployment. Finally, **the team implements mitigation strategies**, including generating *Sigma* detection rules and prioritizing patches for vulnerable *Exchange servers*.

trieval by amplifying irrelevant co-occurrences, leading to misguided contextualization and faulty attribution. Similarly, contradictory knowledge weakens entity recognition and event extraction, causing inconsistent mappings of the same threat actor or IOC across conflicting sources and disrupting downstream attack-chain construction. By connecting each failure mode to the specific LLM workflows used across CTI stages, our analysis provides a detailed account of both their root causes and how they propagate through the reasoning process.

IV. Validation and defensive strategies. We further validate the identified root causes through a causal intervention study, showing that reasoning-time interventions can yield immediate performance gains. Building on these findings, we design and evaluate targeted mitigation strategies, including retrieval-based CTI knowledge filtering and inference-time reasoning constraints. Our results show that these lightweight, taxonomy-informed defenses consistently reduce LLM failures, demonstrating that robust LLM-based CTI analysis requires protections explicitly tailored to the heterogeneity of the threat landscape.

In summary, this work uncovers previously underexplored blind spots in LLM-based CTI systems. We provide foundational insights into their failure modes and initial defensive strategies that can inform more principled adaptations of future CTI-focused LLM systems. To facilitate follow-up research, we release our code at `https://anonymous.4open.science/r/CTI-Gap-3CBA`.

2 Background and Motivation

This section first outlines the scope of CTI tasks (§2.1), then introduces the datasets used to assess LLM performance along with extensive evaluation results (§2.2). These results motivate a deeper investigation into specific failure modes and their root causes, which we address through our methodological design to categorize failure instances (§3) and research findings (§4).

2.1 Background: Stages of CTI Tasks

Cyber threat intelligence (CTI) covers a broad range of cybersecurity activities that support the analysis of threat events and the recommendation of timely, informed incident response.

As illustrated by the **example above**, CTI practices are typically organized into a four-stage pipeline, with each stage involving distinct reasoning tasks and technical methods:

❶ **Contextualization**: Security teams enrich raw observations (e.g., suspicious logs, network alerts, and isolated IOCs) with contextual information to make them actionable. This includes mapping events to known identifiers such as CVEs or MITRE ATT&CK TTPs, linking indicators to malware families, and reconstructing adversarial timelines. **Evaluated CTI techniques** include topic modeling to group related threat narratives, event extraction to identify structured incidents, knowledge base mapping to align observations with established taxonomies, and information retrieval to ground outputs in relevant threat reports or databases.

❷ **Attribution**: Once threats are contextualized, security teams investigate the likely adversaries behind the activity. Attribution links observed events to threat actors or campaigns by analyzing shared TTPs, infrastructure reuse (e.g., overlapping IPs or domains), and stylistic patterns such as language use or operational cadence. **Evalueted CTI techniques** include named entity recognition (NER) to extract actors, malware, and victim entities; relation extraction to identify links among entities and events; structured event graph construction to represent behavioral sequences; and threat actor classification using historical behavioral profiles.

Table 1: Collected cybersecurity benchmarks for LLM-CTI.

Benchmark	Description	Task Coverage				Unique Feature
		❶ Cont.	❷ Attr.	❸ Pred.	❹ Miti.	
CTIBench (Alam et al., 2024)	General CTI tasks	✓	✓	✓	✗	Multi-choice questions (MCQ)
SevenLLM-Bench (Ji et al., 2024)	Report understanding	✓	✓	✓	✓	Synthetic instances, MCQ, QA
SWE-Bench (Jimenez et al., 2023)	Software bug fixing	✗	✓	✗	✓	Program analysis & patching
CyberTeam (Liu et al., 2025)	Blue-team threat hunting	✓	✓	✓	✓	Open-ended decision-making

❸ **Prediction**: With a clearer understanding of the adversary, security teams forecast future threats, exploitation likelihood, and potential impact. This includes estimating the likelihood that known threats will be exploited (e.g., EPSS scoring), anticipating likely target sectors or systems, and modeling campaign evolution. **Evaluated CTI techniques** include historical event correlation to identify temporal patterns, temporal modeling to capture threat progression, and forecasting with time-series or graph-based models to estimate propagation and escalation risks.

❹ **Mitigation**: Finally, CTI supports actionable decisions that reduce risk and guide incident response. This includes recommending patches, tuning detection signatures (e.g., YARA/Sigma rules), adjusting firewall or access-control configurations, and drafting response playbooks. **Evaluated CTI techniques** include mitigation mapping to associate observed TTPs or vulnerabilities with defensive actions, mitigation efficacy prediction to rank response options, and summarization to generate concise, structured remediation steps tailored to the target environment.

Appendix D provides additional introductions on the evaluated CTI techniques, while Appendix G.3 analyzes root causes of LLMs' failure modes triggered by these techniques across CTI stages.

2.2 Motivation: LLMs remain insufficient across diverse CTI tasks

Evaluation Datasets. To evaluate LLM performance on CTI tasks, we leverage benchmarks (Alam et al., 2024; Ji et al., 2024; Jimenez et al., 2023; Liu et al., 2025) as well as real-world threat databases and platforms. The benchmarks are summarized in Table 1, which provide a broad coverage across all CTI stages and capture both structured (MCQ, QA) and unstructured (decision-making, patch generation) task formats. To unify their use, we standardize each instance into a CTI-oriented scenario explicitly aligned with one of the four CTI stages. For example, multi-choice questions (MCQs) from CTIBench and SevenLLM-Bench are reformulated into concrete threat hunting scenarios, such as analyzing a suspicious log entry to determine the relevant TTP or linking an IOC to a known malware family. Our preprocessing mitigates structural biases that could otherwise inflate LLM performance.

We provide the details of real-world databases (or platforms) in Appendix E.2, along with data statistics in Table 7. The prompt template used in evaluation is also included in Appendix E.6.

Evaluated LLMs. Our evaluation covers two complementary lines of models. First, we include industry-leading, general-purpose LLMs (e.g., GPT-5, Claude-Sonnet-4), which represent state-of-the-art reasoning capabilities across domains. Second, we evaluate on open-source or API-accessible cybersecurity-specialized models (e.g., SecGPT (clouditera, 2025), DeepHat (DeepHat, 2025)), which are adapted to security operations through domain-specific training and curated CTI corpora. Appendix E.5 details all evaluated models and their versions.

Model-specific Gap. Across a broad range of CTI tasks (see E.3), Table 2 reveals clear model-specific performance gaps. General-purpose LLMs tend to dominate understanding-intensive and synthesis tasks, whereas cyber-specialized agents show advantages on selected semantic or operational tasks. In contextualization, general models achieve substantial gains, with *Affected Systems* F1 reaching $\sim$82-88% compared with $\sim$55-56% for most cyber agents, and *Source Reliability* AUC reaching $\sim$91% versus a best cyber score of $\sim$74%. Similar trends appear in attribution and prediction, where general LLMs lead on tasks such as *Threat Actor Linking* and *Exploit Likelihood*. However, domain specialization can reverse the gap on certain tasks: cyber agents outperform on *Infrastructure Reuse* and show stronger performance on operational ranking tasks such as *Countermeasure Ranking*. Performance among cyber agents is also notably uneven, suggesting sensitivity to the quality and coverage of domain-specific training data.

Universal Gap. We also observed universal gaps among all models: limited IOC normalization and CVE linking under obfuscation, inconsistent TTP extraction across retrieval-augmented reasoning,

Table 2: Evaluation (mean±std across five runs) of LLMs on CTI tasks across four CTI stages. Industry-leading general-purpose LLMs (left) are compared with cybersecurity-specialized models (right). Detailed CTI task descriptions (with examples) are provided in E.3. Model names and versions are introduced in E.5. Smaller gray values in parentheses denote standard deviations.

Detailed CTI Task	Metric	Industry-leading, general-purpose						Cybersecurity-specialized					
		G5	CLD	GEM	LL70	MIX	QWN	FSC	ZYS	LLY	CBS	SPT	DHT
❶ Contextualization													
Affected Systems	F1	82.2 (3.0)	75.7 (9.2)	61.3 (1.7)	88.2 (0.4)	66.3 (1.9)	74.7 (0.7)	43.2 (0.5)	55.3 (2.1)	56.6 (0.9)	56.2 (8.0)	55.4 (3.0)	55.9 (4.5)
Attack Infrastructure	F1	86.3 (8.3)	61.4 (1.1)	57.8 (4.9)	85.3 (6.2)	62.8 (3.9)	61.6 (0.7)	50.7 (4.9)	61.2 (2.1)	62.5 (10.9)	61.8 (4.3)	61.1 (0.8)	61.6 (2.0)
Vulnerability Linking	Acc	65.2 (6.1)	60.2 (5.0)	57.4 (2.8)	64.9 (2.4)	53.3 (3.2)	52.1 (4.5)	51.2 (2.3)	62.1 (3.9)	63.3 (3.5)	62.9 (4.4)	62.2 (3.6)	62.8 (2.7)
Malware Family Mapping	F1	68.1 (1.1)	63.5 (1.0)	60.2 (0.8)	58.4 (2.9)	56.7 (3.1)	55.1 (5.1)	54.1 (8.3)	63.9 (0.5)	65.2 (9.8)	64.6 (0.9)	64.1 (4.4)	64.8 (2.5)
IOC Normalization	F1	72.1 (3.6)	68.2 (4.9)	66.1 (3.3)	64.2 (3.0)	62.3 (5.1)	60.9 (3.2)	60.2 (0.2)	67.8 (8.8)	68.9 (2.6)	68.4 (2.5)	67.8 (0.9)	68.3 (1.3)
Threat Report Alignment	BLEU	42.9 (5.9)	20.6 (1.9)	39.6 (8.6)	48.6 (3.9)	27.9 (1.7)	27.1 (3.6)	15.9 (6.5)	44.1 (3.0)	34.8 (2.4)	24.4 (2.7)	13.9 (1.9)	34.2 (4.2)
Event Timeline Construction	BLEU	56.3 (2.1)	53.2 (1.9)	51.9 (2.3)	50.4 (2.2)	49.2 (3.1)	47.8 (1.7)	47.2 (3.4)	59.4 (2.4)	60.2 (9.8)	59.9 (4.5)	59.3 (0.3)	59.7 (3.5)
Graph Population	Acc	79.3 (1.8)	55.9 (2.7)	53.9 (3.0)	72.4 (3.1)	50.7 (5.8)	69.3 (2.2)	48.7 (9.0)	42.1 (2.7)	33.4 (1.6)	62.9 (7.6)	42.4 (2.6)	62.8 (2.4)
Source Reliability Scoring	AUC	91.2 (2.1)	77.3 (10.5)	86.1 (2.9)	74.5 (6.3)	63.1 (3.2)	71.7 (7.0)	71.2 (6.4)	73.8 (1.4)	54.7 (0.9)	64.2 (2.7)	53.7 (3.3)	64.1 (2.2)
❷ Attribution													
Threat Actor Linking	Acc	89.2 (1.0)	65.2 (2.5)	82.2 (1.0)	59.8 (3.3)	77.3 (2.1)	75.3 (1.4)	64.3 (0.2)	70.4 (0.8)	41.3 (2.1)	70.7 (2.6)	77.1 (2.9)	60.8 (5.1)
TTP Extraction	F1	75.1 (1.6)	72.4 (5.1)	70.3 (1.6)	47.8 (3.4)	66.9 (1.8)	65.4 (4.0)	65.4 (7.2)	72.4 (6.1)	53.7 (4.7)	73.1 (5.1)	72.6 (1.9)	73.2 (4.1)
Campaign Attribution	Acc	71.2 (4.6)	67.1 (0.4)	64.9 (1.3)	63.1 (1.9)	60.7 (1.6)	58.6 (0.8)	57.8 (2.9)	69.4 (3.0)	70.3 (2.5)	69.9 (2.2)	69.4 (5.9)	70.1 (3.0)
Infrastructure Reuse	F1	67.7 (9.1)	63.6 (9.9)	60.9 (3.2)	59.1 (4.2)	57.4 (3.7)	55.6 (2.2)	54.8 (1.2)	68.8 (4.7)	52.8 (3.3)	69.2 (3.0)	75.4 (0.9)	60.3 (3.0)
Language/Style Profiling	Acc	59.8 (5.2)	56.1 (1.0)	53.9 (3.1)	52.1 (8.0)	50.3 (2.3)	48.8 (1.8)	48.9 (5.6)	63.2 (10.4)	64.3 (1.0)	63.8 (9.0)	63.3 (0.8)	63.9 (3.4)
False Flag Detection	F1	67.9 (0.5)	50.1 (0.6)	48.6 (11.5)	67.2 (5.0)	45.9 (1.2)	54.7 (5.6)	44.4 (7.5)	57.4 (1.2)	28.6 (1.9)	46.2 (3.1)	57.6 (2.5)	58.2 (2.1)
Evidence Weighting	BLEU	36.2 (2.4)	13.1 (6.4)	22.6 (2.9)	40.2 (3.2)	28.8 (0.5)	17.8 (4.4)	7.3 (2.9)	9.7 (2.0)	0.7 (0.4)	10.2 (1.2)	19.7 (1.3)	8.3 (2.8)
Relation Graph Building	F1	64.2 (3.1)	61.1 (4.7)	59.5 (3.7)	57.9 (2.1)	56.2 (4.6)	54.7 (3.2)	54.4 (3.1)	67.5 (7.7)	68.3 (9.0)	67.8 (11.0)	67.3 (3.9)	67.9 (2.0)
❸ Prediction													
Exploit Likelihood	AUC	82.1 (8.6)	79.2 (6.8)	77.1 (8.8)	85.6 (2.0)	74.2 (2.3)	62.9 (6.1)	51.9 (2.6)	74.2 (9.1)	55.9 (1.1)	76.4 (1.9)	65.3 (8.6)	75.9 (4.7)
Impact Forecast	BLEU	49.8 (2.5)	27.1 (2.3)	35.4 (4.2)	44.1 (1.9)	22.6 (2.5)	21.3 (3.9)	10.8 (3.7)	20.7 (5.1)	11.9 (3.1)	4.6 (3.2)	11.5 (1.8)	22.1 (0.9)
Target Sector Prediction	Acc	84.1 (2.6)	74.3 (2.3)	81.9 (1.4)	60.2 (0.2)	75.6 (1.9)	71.2 (3.5)	56.4 (2.1)	50.2 (8.8)	31.1 (2.3)	62.3 (9.2)	41.5 (6.5)	61.9 (2.0)
Campaign Escalation	AUC	68.3 (3.1)	61.1 (1.2)	59.8 (3.9)	58.2 (1.1)	56.8 (0.9)	55.4 (1.4)	54.8 (6.5)	60.7 (2.1)	51.8 (1.4)	65.2 (2.9)	60.7 (8.4)	61.3 (6.3)
❹ Mitigation													
Patch Recommendation	F1	70.2 (3.0)	65.9 (3.2)	63.6 (2.2)	71.8 (10.9)	60.1 (2.2)	58.3 (8.8)	58.2 (1.1)	63.2 (0.5)	44.2 (3.1)	62.9 (9.6)	64.1 (3.3)	44.6 (3.9)
Rule Generation (YARA)	BLEU	38.2 (7.3)	33.9 (3.0)	48.2 (3.1)	26.7 (3.5)	21.3 (2.5)	33.7 (1.7)	23.1 (9.6)	28.1 (4.8)	8.9 (3.3)	9.4 (0.6)	28.7 (3.2)	20.2 (4.1)
Response Summarization	BLEU	51.4 (1.5)	28.6 (1.3)	56.7 (2.9)	25.2 (3.8)	23.6 (4.5)	42.2 (4.8)	11.8 (3.2)	31.5 (3.7)	22.7 (0.7)	43.3 (5.8)	12.6 (3.7)	20.8 (6.1)
Mitigation–TTP Mapping	Acc	67.2 (2.7)	63.3 (6.7)	61.1 (2.7)	59.6 (9.5)	57.8 (1.4)	56.1 (2.9)	55.9 (1.8)	61.3 (1.7)	62.6 (2.0)	63.1 (7.7)	62.4 (2.0)	62.9 (0.9)
Defensive Playbook Gen	BLEU	58.6 (5.1)	55.7 (2.6)	53.7 (2.3)	52.2 (1.9)	50.8 (3.5)	49.5 (3.0)	49.1 (4.7)	56.1 (3.3)	57.2 (3.1)	57.6 (1.9)	57.1 (3.0)	57.5 (4.4)
Countermeasure Ranking	NDCG	59.1 (8.8)	56.1 (8.0)	54.7 (3.3)	53.2 (5.0)	51.9 (2.6)	50.4 (1.2)	50.3 (5.0)	62.3 (2.0)	53.3 (2.7)	62.9 (1.8)	42.4 (3.2)	42.9 (4.1)
Incident Ticket Generation	Acc	83.1 (2.6)	60.1 (2.7)	68.2 (3.3)	86.8 (3.4)	75.3 (2.7)	63.9 (3.5)	53.6 (1.7)	65.3 (1.6)	46.4 (9.7)	66.8 (9.3)	60.2 (2.2)	56.4 (0.7)

and weak temporal coherence in timelines/escalation forecasts. We also observe shallow reliance on real-world evidence in report alignment (low BLEU) and format errors in ticket/playbook generation. These deficiencies present with both larger online models (e.g., GPT-5) and cyber-specific agents, implying inherent limitations that may be triggered by the nature of the threat landscape.

Motivated by these gaps, we investigate latent **failure modes** of LLMs that impede effective decision-making among CTI tasks. We aim to build a systematic view about where models break down and why these failure modes persist despite large-scale pretraining or domain-specific adaptation.

3 Methodology: Categorize Failure Instances

After conducting the large-scale evaluation described in §2, we categorize the resulting failure cases to better understand the limitations of LLMs. This process is guided by three research questions:

RQ_1: How can we identify "failure" especially in tasks that lack hard-label annotations?
RQ_2: How can we determine the finite scope of failure modes?
RQ_3: How can we efficiently categorize large-scale instances into these failure modes?

Overall, **we do not fully trust LLM-as-judge or AI-produced confidence** to detect failure cases, due to the lack of transparency and the risk of "self-rationalized" reasoning. Nevertheless, the large scale of the evaluation set requires us to categorize instances efficiently beyond purely manual efforts. To address this, we propose a stratification approach that partitions instances based on their "failure" depth, followed by an autoregressive method combined with manual efforts to resolve the above RQs. Figure 2 illustrates the overall workflow.

3.1 Stratifying instances without hard-label annotations (RQ_1)

Most cyber threat intelligence tasks, such as mitigation rule generation, lack binary hard labels. However, authoritative reference materials (e.g., CVE (CVE Program, 2024) or advisories from CISA (Cybersecurity and Infrastructure Security Agency (CISA), 2024)) allow us to leverage reference-based similarity metrics (e.g., BLEU) to quantify the alignment between model outputs and ground truth.

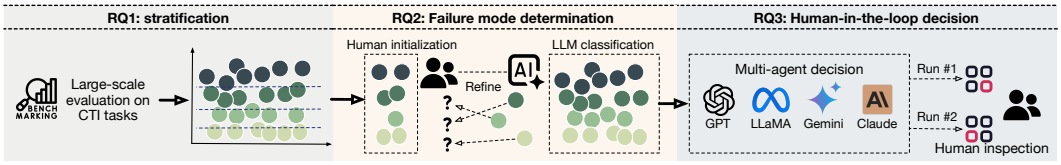


Figure 2: Overview of method to categorize failure instances (addressing RQ_1-RQ_3).

To efficiently identify failure cases, we design a stratification approach (Algorithm 1). We first compute similarity scores for all instances and partition them into 5% quantile-based strata. Within each stratum, we manually inspect a small set of anchor samples to establish score ranges for "correct" versus "failed" outputs. Remaining instances in the stratum are then automatically labeled based on these ranges; boundary cases falling in the overlapping region will trigger additional review. This process proceeds iteratively through the strata and terminates once the distribution of "failed" instances stabilizes and no new sets found. Practically, this hybrid approach yields high-confidence failure sets while requiring manual inspection of less than 1.5% of the total instances.

3.2 Failure mode determination (RQ_2)

We avoid relying on an LLM-only judge to determine failure modes, instead, we design an **iterative** process alternating between light-weight human annotation and LLM-assisted classification. Let $\mathcal{D} = \{x_i\}_{i=1}^{N}$ denote the set of failure instances (from stratification results in §3.1).

Step 1 (Initialization). Human annotators randomly inspect a small subset $\mathcal{D}_0 \subset \mathcal{D}$ to derive an initial taxonomy of failure modes, $\mathcal{T}_0 = t_1, \ldots, t_k$.

Step 2 (LLM classification). For each remaining instance $x_j \in \mathcal{D} \setminus \mathcal{D}_0$, an LLM assigns a label $y_j \in \mathcal{T}_m \cup \{\texttt{other}\}$, where $\mathcal{T}_m$ is the taxonomy after m iterations.

Step 3 (Refinement). Instances labeled as `other`, i.e., $\mathcal{O}_m = \{x_j \mid y_j = \texttt{other}\}$, are further inspected by human annotators. If new failure patterns are identified, new modes $\Delta\mathcal{T}$ are added to the taxonomy, yielding $\mathcal{T}_{m+1} = \mathcal{T}_m \cup \Delta\mathcal{T}$.

This loop repeats until $\Delta\mathcal{T} = \varnothing$, i.e., no new failure modes are found. We ultimately find the stabilized set of failure modes: $\mathcal{T}^* = \lim_{m\to\infty} \mathcal{T}_m$ (see Algorithm 2). Note that determining failure modes requires comparing failure cases with the ground-truth answers (or references). The specific methods tailored to each failure mode type are detailed in Appendix F.3.

3.3 Human-in-the-loop decision (RQ_3)

To balance reliability and scalability in large-scale categorization, we integrate human inspection with multi-agent LLM decisions:

Step 1 (Multi-agent decision). For each threat instance x_i requiring failure mode classification, we construct a model set $\Theta = \{\text{GPT-5}, \text{Llama-4-17B}, \text{Gemini-2.5}, \text{Claude-Sonnet-4}\}$. Each model independently proposes an initial label $\hat{y}_i^{(1)} \in \mathcal{T}$ during the first execution.

Step 2 (Repetition for stability). Motivated by evidence that unstable internal knowledge may lead to fluctuations across runs (Kumar et al., 2024), we execute a second round of multi-agent deliberation. In this round, each LLM observes the predictions $\hat{y}_i^{(1)}$ from other models, refines its reasoning, and then proposes a new label $\hat{y}_i^{(2)}$ for the same instance.

Step 3 (Human verification). Human inspectors evaluate consistency along two dimensions: (1) if agent-level votes show disagreement (lack of consensus), or (2) if $\hat{y}_i^{(1,a)} \neq \hat{y}_i^{(2,a)}$ (fluctuation across rounds by any agent a). In either case, the instance is flagged as uncertain:

$$U = \left\{ x_i : \left| \{\hat{y}_i^{(1,a)} : a \in \Theta\} \right| > 1 \ \vee \ \exists a,\ \hat{y}_i^{(1,a)} \neq \hat{y}_i^{(2,a)} \right\}$$

where $\{\hat{y_i}^{(a)}\}$ denotes the set of labels assigned by agents $a \in \Theta$. All $x_i \in \mathcal{U}$ are then inspected by human annotators for final determination.

Table 3: Distribution (ratio) of categorized CTI failure modes across different methods. **Given the absence of ground truth (hard labels), human annotation serves as a near-gold standard. Performance closely matching human annotation demonstrates a near-human capability for failure discovery.** Smaller gray values in parentheses denote standard deviations.

Method	Spurious Correlation					Contradictory Knowledge						Constrained Generalization			
	1.1 Co-men. bias	**1.2** Exploit. bias	**1.3** Confond. factor	**1.4** Skewed source	**1.5** Hier. metadata	**2.1** Temporal contra.	**2.2** Conflict report	**2.3** Semantic conflict	**2.4** Diver. data	**2.5** Misal. knowl.	**2.6** Counter. CTI	**3.1** Distribut. bias	**3.2** Unseen pattern	**3.3** Overfit. reasoning	**3.4** Env. unaware
Human annotation	4.14 (0.20)	1.71 (0.11)	15.72 (3.04)	6.16 (0.81)	2.05 (0.08)	13.98 (1.31)	22.29 (1.14)	4.14 (0.86)	4.53 (0.59)	1.97 (0.19)	7.45 (0.80)	21.79 (2.29)	23.13 (4.30)	7.64 (1.09)	11.76 (2.54)
Clustering	8.53 (0.52)	5.24 (1.44)	10.16 (1.12)	12.41 (0.37)	6.83 (0.26)	18.37 (2.59)	15.68 (1.71)	10.21 (0.26)	9.87 (1.16)	6.54 (0.29)	14.26 (1.78)	18.59 (0.71)	16.42 (3.64)	15.84 (0.57)	16.23 (1.35)
Topic modeling	2.19 (0.56)	0.57 (0.11)	22.32 (0.55)	3.14 (0.18)	0.42 (0.62)	8.43 (1.48)	30.12 (4.72)	1.28 (0.50)	2.34 (0.01)	0.69 (0.55)	3.52 (0.32)	32.41 (1.53)	12.57 (1.00)	2.16 (0.43)	4.89 (1.18)
GPT judge	10.27 (0.05)	7.13 (0.44)	12.59 (1.76)	8.92 (0.90)	5.37 (0.04)	10.51 (0.26)	19.84 (4.57)	8.46 (0.19)	8.12 (0.06)	5.58 (0.11)	11.23 (0.19)	20.17 (3.04)	18.29 (2.47)	14.38 (0.90)	13.94 (1.00)
Gemini judge	5.82 (2.97)	3.42 (0.84)	14.23 (1.11)	7.56 (0.31)	3.29 (0.57)	15.19 (0.37)	21.07 (0.64)	5.53 (0.64)	5.86 (0.29)	3.13 (0.76)	9.47 (0.90)	21.54 (2.31)	26.83 (2.43)	11.27 (0.61)	11.62 (2.19)
Multi-agent debate	3.91 (0.23)	4.18 (0.29)	16.87 (0.95)	5.29 (0.87)	4.81 (0.52)	12.24 (3.03)	24.53 (2.43)	6.89 (1.53)	3.27 (0.16)	4.62 (0.31)	6.18 (0.15)	25.46 (3.90)	20.61 (4.79)	12.42 (1.67)	8.57 (1.72)
Ours	4.12 (0.02)	1.87 (0.28)	15.50 (0.48)	6.31 (0.95)	1.98 (0.07)	13.86 (0.58)	22.33 (1.27)	3.96 (0.76)	4.49 (1.37)	1.85 (0.09)	7.77 (0.87)	23.34 (1.25)	25.06 (2.35)	10.02 (1.17)	10.80 (0.89)

The multi-agent collaboration efficiently handles most routine cases , while humans focus only on instances with instability or misalignment (empirically, fewer than 1.8% cases in this step), thus balancing scalability with reliability in failure mode categorization (Algorithm 3).

4 Experimental Result

Our categorization method leads to a detailed analysis of failure mode classes, as presented in Figure 1. Building on this, we further investigate specific failure cases under the same settings as in §2.2.

4.1 Effectiveness of failure mode categorization with baselines

Baseline setting. To validate the efficacy of our proposed framework, we benchmark it against several basic methodologies: **(i) Clustering** aggregates evaluation results of CTI benchmarks (§2.2) into unsupervised groups based on vector embedding similarity using *K-Means*. **(ii) Topic modeling** leverages latent dirichlet allocation to statistically extract semantic structures and keywords from the benchmarking results (§2.2). **(iii) GPT judge**, wherein GPT-5 independently classifies failure modes based on provided definitions. **(iv) Gemini judge** similarly leverages the Gemini model to categorize failure modes. **(v) Multi-agent debate** constructs an iterative consensus mechanism where multiple LLM agents (same as in §3.3) refine each other's classifications to reduce individual hallucinations.

Results. The comparison in Table 3 shows clear gaps between unsupervised baselines and semantic reasoning approaches. Statistical methods such as clustering and topic modeling miss many of the fine-grained subtype boundaries, while producing a skewed failure distribution that departs substantially from the human reference. LLM-based judges align more closely with human judgments, but still exhibit drift on the more challenging categories that require multi-step reasoning. Overall, our method yields a subtype distribution that is consistently closest to human annotation, correcting the systematic skews seen in both single-agent predictions and simple consensus-style debates.

Explanations. The remaining gap largely reflects how well each method handles context-dependent ambiguity. Topic modeling, for example, struggles with *Confounding factors (1.3)* and *Temporal contradictions (2.1)* because these errors are driven by latent causal structure and logical inconsistency rather than surface-level lexical similarity. Standard LLM judges reduce this mismatch, but they often over-attribute cases to *Spurious correlations (1.x)* and inflate *Overfitted reasoning (3.3)*, effectively "finding" patterns that are plausible in the abstract yet unsupported by the specific incident context. Our approach mitigates these tendencies by grounding decisions in explicitly retrieved evidence and structured multi-agent deliberation, which improves attribution for high-nuance categories such as *Conflicting reports (2.2)* and *Unseen patterns (3.2)* that other methods frequently misclassify.

Due to space limitations, we defer the (1) ablation study to Appendix G.1, and (2) the results of LLM performance across different failure modes to Appendix G.2.

4.2 Root cause analysis

During failure categorization (§3), we gained insight into how different failure modes are triggered by the techniques used across CTI tasks. Table 4 summarizes these drivers, aligned with the failure mode types in Figure 1. Below, we highlight a representative case of analysis.

Table 4: Summarized root causes of failure modes, with detailed analysis in Appendix G.3.

Category	Subtype	Stage	How Failure Happens
Spurious Correlation	(1.1) Co-mention bias	❶❷❸❹	RAG leverages unrelated but co-mentioned mitigations or cyber threats, causing LLMs to infer false associations.
	(1.2) Exploitation bias	❶❷❸❹	Reused IoCs across incidents mislead models into over-attributing infrastructure reuse or ongoing activity.
	(1.3) Confounding factors	❶❷❸❹	Non-causal variables (e.g., geography, org tags) treated as causal predictors for exploit likelihood.
	(1.4) Skewed source	❶	Overrepresentation of certain feeds biases patch prioritization and defensive rules toward specific vendors.
	(1.5) Hierarchical metadata	❶❷❸	Structured taxonomies (e.g., ATT&CK chains) interpreted as causal orderings rather than descriptive metadata.
Contradictory Knowledge	(2.1) Temporal contradiction	❶❷❸❹	Outdated advisories conflict with newer reports, confusing model reasoning about valid mitigations.
	(2.2) Conflicting reports	❶❷❸❹	Disagreement between sources on documented attack context, actor, dependencies, or other evidence.
	(2.3) Semantic conflict	❶	Different naming/taxonomies (e.g., PlugX vs. Korplug) cause inconsistency.
	(2.4) Divergent structures	❶❷❸	JSON feeds vs. unstructured PDFs produce inconsistencies when fused into CTI mapping (e.g., TTP to patch).
	(2.5) Misaligned standards	❷❸❹	Differences in scoring frameworks (CVSS vs. vendor ratings) yield contradictory threat intelligence.
	(2.6) Counteracting generation	❷❸❹	Reasoning and generation on CTI tasks disrupt LLM safety alignment, causing unstable outputs.
Constrained Generalization	(3.1) Distributional bias	❶	Training on limited language/region corpora hinders generalization to unseen threat contexts.
	(3.2) Unseen patterns	❷❸❹	Zero-day exploits exhibit novel paths absent from model training, degrading exploitability forecasts.
	(3.3) Overfitted reasoning	❷❸	Memorized patterns (e.g., CVE-TTP) lead to brittle linking and ineffective generation.
	(3.4) Environmental unawareness	❷❸❹	Models overlook local system/sector-specific dependencies, producing ineffective mitigation strategies.

Case Study I: co-mention bias (Type 1.1) in ❶ contextualization

Microsoft Patch Tuesday advisories often list many CVEs under one product (e.g., *Windows Server*) (Ruotsalainen, 2018). The contextualization pipeline ingests the bulletin and a threat report cites *CVE-2021-34527 (PrintNightmare)*, GPT-5 incorrectly infers that all co-mentioned CVEs were exploited in the same campaign, creating false links between irrelevant CVEs and active threats.

Root cause of failure mode within a single CTI stage: we first conduct failure-modes-specific inspections across different CTI stages, investigating why a failure mode may (or may not) arise in a given stage. Here is one example of **co-mention bias (Type 1.1) in ❶ contextualization:** As shown in Case Study I, co-mention bias arises during contextualization when retrieval systems treat entities that co-occur in raw reports as if they were causally related. Security bulletins often list indicators or mitigations in a single section, even though only a subset is relevant to the incident at hand. Methods such as topic modeling or coarse knowledge-base mapping, which lack fine-grained disambiguation, may surface all co-mentioned entities as equally relevant. This bloats the retrieved context with spurious links (e.g., attaching unrelated CVEs or MITRE TTPs to the same intrusion), which in turn prompts LLMs to propagate incorrect associations.

Case Study II: The intertwined threats: MOVEit (CVE-2023-34362)

Early CTI reporting showed **contextualization failure (Type 1.3)**: triage lumped the SQL injection into generic IIS exploitation, and *co-mention bias* hid the specific bug in `moveitisapi.dll`. Later, **Attribution failure (Type 2.1)** misattributed the activity to *Silence* due to the "Truebot" downloader, despite later evidence pointing solely to *Cl0p (Lace Tempest)*. Finally, **Prediction failure (Type 3.2)** underestimated the *Impact Forecast*, missing the supply-chain cascade (e.g., Zellis, BBC) because the model overgeneralized from typical single-victim ransomware cases.

Moreover, we also observed **intertwined failure modes**. As shown in Case Study II, we also observe that multiple failure modes may interact within the same CTI instance. This stems from the tightly coupled nature of CTI pipelines: early-stage retrieval issues (e.g., co-mention bias or source skew) are carried forward as "facts," leaving downstream attribution and prediction models with little room to falsify them. The problem is further amplified by heterogeneous and drifting evidence that forces LLMs to memorize signals that are not truly compatible. Under this uncertainty, models fall back on inductive shortcuts (e.g., memorized actor-TTP associations), turning partial or noisy metadata into brittle reasoning. Finally, because CTI reasoning is hierarchical and dependency-driven (e.g., attack chains and mitigation mappings), misreading one link can cascade into adjacent stages. As a result, failure mode combinations such as co-mention bias paired with temporal contradiction, or unseen patterns coupled with environmental unawareness, can form self-reinforcing loops that entangle multiple failures within a single instance.

A comprehensive analysis with case studies across all failure mode types is provided in Appendix G.3. Appendix G.4 further provides additional analyses on intertwined failures.

4.3 Defensive Strategies

Inference-time defense. Our failure analysis reveals a shared pattern across different failure modes: LLM reasoning can depend excessively on incidental context instead of threat attributes that should determine the CTI judgment. We therefore introduce a lightweight inference-time **counterfactual**

consistency check (C^3) defensive strategy. Specifically: given an input x, we construct counterfactual variants $x'_k = I_k(x)$ by perturbing non-causal context (e.g., dates or reporting artifacts) while preserving causal threat attributes such as exploitation mechanisms and malware behaviors. A robust prediction should remain stable under these interventions. We quantify this stability as $\Omega(y|x) = \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} \mathbb{1}\left[\arg\max_{y'} P(y'|x'_k) = y\right]$, where a low Ω indicates sensitivity to non-causal evidence and thus a potentially vulnerable prediction. C^3 also corrects them by aggregating predictions over the counterfactual set: $\hat{y} = \arg\max_y \sum_{k\in\mathcal{K}} w_k P(y|x'_k)$, where w_k downweights uncertain reasoning paths. This encourages predictions supported by evidence that remains invariant across contextual perturbations.

As previewed in Table 5, the intervention consistently improves representative tasks across all four CTI stages, with particularly large gains on evidence-sensitive tasks such as *False Flag Detection* and *Threat Report Alignment*. Comprehensive results across all models and tasks, together with implementation details and fine-grained mitigation strategies, are provided in Appendix H.

Table 5: Representative absolute gains (Δ) from C^3. Full results are in Appendix H.

CTI Task	Metric	G5	CLD	MIX	FSC
Threat Report Alignment	BLEU	+8.2	+11.2	+10.5	+9.1
False Flag Detection	F1	+8.8	+10.8	+11.5	+9.5
Impact Forecast	BLEU	+6.5	+9.2	+9.5	+7.5
Defensive Playbook Gen.	BLEU	+6.2	+8.8	+9.2	+7.2

5 Related Work

LLM-as-agent. LLM-based agents have been explored across diverse domains such as education (Tang et al., 2025; Chu et al., 2025; Neumann et al., 2024), scientific discovery (Shojaee et al., 2024; Ren et al., 2025), healthcare (Goyal et al., 2024; Zhou et al., 2024a; More et al., 2026; Meng et al., 2026; Brens et al., 2026), and urban mobility modeling (Zhou et al., 2024b; Wang et al., 2024; Calderón et al., 2025). These applications highlight the ability of LLM agents to decompose complex tasks, integrate external tools, and generate executable outputs.

LLMs for cybersecurity. LLMs are increasingly leveraged for both offensive and defensive cybersecurity tasks, owing to their strong natural language understanding and reasoning capabilities. In static analysis, LSAST augments traditional SAST tools with dynamic vulnerability knowledge (Keltek et al., 2025), while hybrid systems use LLM-driven preprocessing and explanation to improve anomaly detection in IoT (Ghimire et al., 2025). Autonomous agents have demonstrated the ability to exploit one-day vulnerabilities (Fang et al., 2024) and guide fuzzing across multi-hop library dependencies (Zhou et al., 2024c). Additional applications include intrusion detection (G. Lira et al., 2024), large-scale code review (Sun et al., 2025), and malware tracking (Yu et al., 2024).

Vulnerabilities of LLMs. Despite their strong capabilities, LLM-based agents exhibit critical vulnerabilities that undermine reliability in high-stakes domains. Recent work shows that malfunction amplification attacks can cascade small reasoning errors into severe misjudgments (Zhang et al., 2025). Other studies highlight vulnerabilities to data poisoning (Wang et al., 2025), adversarial prompts (Debenedetti et al., 2024), and misalignment (Fang et al., 2025), all raise concerns for deploying LLMs in high-stakes environments.

6 Conclusion

This work presents a systematic study of intrinsic failure modes that constrain the effectiveness of LLMs in cyber threat intelligence. We combine large-scale benchmark evaluations with real-world CTI reports and uncover three fundamental failure modes that persist across multiple CTI stages and workflows. Our developed methodology enables reliable categorization of failure instances and provides insights into how these limitations emerge and propagate. Correspondingly, we propose targeted mitigation strategies for different failure-mode subtypes, with further evaluations demonstrating their effectiveness. Overall, our findings reveal blind spots in current cybersecurity-centric LLM reasoning and provide a path toward more principled model adaptation and robust cyber-agent design. **Ethics statement:** Our study uses publicly available cybersecurity benchmarks, threat reports, and technical security data, without collecting private or personally sensitive information. Human involvement is limited to technical annotation of CTI materials by domain-experienced annotators, and no offensive capabilities or previously undisclosed software vulnerabilities are released. The study is intended to improve the reliability and defensive use of LLM-based CTI systems.

REFERENCES

Francisco Aguilera-Martínez and Fernando Berzal. Llm security: Vulnerabilities, attacks, defenses, and countermeasures. *arXiv preprint arXiv:2505.01177*, 2025.

Bader Al-Sada, Alireza Sadighian, and Gabriele Oligeri. Mitre att&ck: State of the art and way forward. *ACM Computing Surveys*, 57(1):1–37, 2024.

Md Tanvirul Alam, Dipkamal Bhusal, Le Nguyen, and Nidhi Rastogi. Ctibench: A benchmark for evaluating llms in cyber threat intelligence. *Advances in Neural Information Processing Systems*, 37:50805–50825, 2024.

Luca Allodi. Economic factors of vulnerability trade and exploitation. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 1483–1499, 2017.

Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

Joakim Berndtsson. Combining semi-structured interviews and document analysis in a study of private security expertise. In *Researching non-state actors in international security*, pp. 81–95. Routledge, 2017.

Leyla Bilge and Tudor Dumitraş. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 833–844, 2012.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Rafael Brens, Yuqiao Meng, Luoxi Tang, and Zhaohan Xi. Semantic nlp pipelines for interoperable patient digital twins from unstructured ehrs. *arXiv preprint arXiv:2601.05847*, 2026.

Obrina Briliyant, Amir Javed, and Yulia Cherdantseva. Network packet security analysis chatbot using sql-rag approach and cybersecurity-tuned llm. In *2025 International Conference on Electrical, Communication and Computer Engineering (ICECCE)*, pp. 1–6. IEEE, 2025.

Benjamin L Bullough, Anna K Yanchenko, Christopher L Smith, and Joseph R Zipkin. Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics*, pp. 45–53, 2017.

Christian Calderón, Pasqual Martí, Jaume Jordán, Javier Palanca, and Vicente Julian. Cognitive agents in urban mobility: Integrating llm reasoning into multi-agent simulations. *Sensors*, 25(18): 5688, 2025.

Shin-Ming Cheng, Weng Chon Ao, Pin-Yu Chen, and Kwang-Cheng Chen. On modeling malware propagation in generalized social networks. *IEEE Communications Letters*, 15(1):25–27, 2010.

Othmane Cherqi, Hicham Hammouchi, Mounir Ghogho, and Houda Benbrahim. Leveraging open threat exchange (otx) to understand spatio-temporal trends of cyber threats: Covid-19 case study. In *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 1–6. IEEE, 2021.

Zhendong Chu, Shen Wang, Jian Xie, Tinghui Zhu, Yibo Yan, Jinheng Ye, Aoxiao Zhong, Xuming Hu, Jing Liang, Philip S Yu, et al. Llm agents for education: Advances and applications. *arXiv preprint arXiv:2503.11733*, 2, 2025.

Vanessa Clairoux-Trepanier, Isa-May Beauchamp, Estelle Ruellan, Masarah Paquet-Clouston, Serge-Olivier Paquette, and Eric Clay. The use of large language models (llm) for cyber threat intelligence (cti) in cybercrime forums. *arXiv preprint arXiv:2408.03354*, 2024.

clouditera. secgpt. `https://huggingface.co/clouditera/secgpt`, 2025.

Zachary Coalson, Jeonghyun Woo, Shiyang Chen, Yu Sun, Lishan Yang, Prashant Nair, Bo Fang, and Sanghyun Hong. Prisonbreak: Jailbreaking large language models with fewer than twenty-five targeted bit-flips. *arXiv e-prints*, pp. arXiv–2412, 2024.

CVE Program. Common vulnerabilities and exposures (cve). https://www.cve.org/, 2024.

CyberNative. Cyberbase-13b. https://huggingface.co/CyberNative/CyberBase-13b, 2024.

Cybersecurity and Infrastructure Security Agency (CISA). Known exploited vulnerabilities catalog. https://www.cisa.gov/known-exploited-vulnerabilities-catalog, 2024.

Tobias Dam and Sebastian Neumaier. Towards measuring vulnerabilities and exposures in open-source packages. In *International Data Science Conference*, pp. 13–19. Springer, 2023.

Francesco De Santis, Kai Huang, Rodolfo Valentim, Danilo Giordano, Marco Mellia, Zied Ben Houidi, and Dario Rossi. Cfa-bench: Cybersecurity forensic llm agent benchmark and testing. In *2025 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 217–225. IEEE, 2025.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

DeepHat. Deephat-v1-7b. https://huggingface.co/DeepHat/DeepHat-V1-7B, 2025.

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.

Adji B Dieng, Francisco JR Ruiz, and David M Blei. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453, 2020.

Xiaohu Du, Ming Wen, Jiahao Zhu, Zifan Xie, Bin Ji, Huijun Liu, Xuanhua Shi, and Hai Jin. Generalization-enhanced code vulnerability detection via multi-task instruction fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10507–10521, 2024.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. Preemptive detection and correction of misaligned actions in llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 222–244, 2025.

Richard Fang, Rohan Bindu, Akul Gupta, and Daniel Kang. Llm agents can autonomously exploit one-day vulnerabilities. *arXiv preprint arXiv:2404.08144*, 2024.

Robert Friel and Atindriyo Sanyal. Chainpoll: A high efficacy method for llm hallucination detection. *arXiv preprint arXiv:2310.18344*, 2023.

Oscar G. Lira, Alberto Marroquin, and Marco Antonio To. Harnessing the advanced capabilities of llm for adaptive intrusion detection systems. In *International Conference on Advanced Information Networking and Applications*, pp. 453–464. Springer, 2024.

Ashutosh Ghimire, Ghazal Ghajari, Karma Gurung, Love K Sah, and Fathi Amsaad. Enhancing cybersecurity in critical infrastructure with llm-assisted explainable iot systems. In *2025 1st International Conference on Secure IoT, Assured and Trusted Computing (SATC)*, pp. 1–5. IEEE, 2025.

Sagar Goyal, Eti Rastogi, Sree Prasanna Rajagopal, Dong Yuan, Fen Zhao, Jai Chintagunta, Gautam Naik, and Jeff Ward. Healai: A healthcare llm for effective medical documentation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 1167–1168, 2024.

Luke Guerdan, Solon Barocas, Kenneth Holstein, Hanna Wallach, Zhiwei Steven Wu, and Alexandra Chouldechova. Validating llm-as-a-judge systems in the absence of gold labels. *arXiv e-prints*, pp. arXiv–2503, 2025.

Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd annual computer security applications conference*, pp. 103–115, 2017.

Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. Exploit prediction scoring system (epss). *Digital Threats: Research and Practice*, 2(3):1–17, 2021.

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.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1827–1843, 2023.

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.

Mete Keltek, Rong Hu, Mohammadreza Fani Sani, and Ziyue Li. Lsast: Enhancing cybersecurity through llm-supported static application security testing. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pp. 166–179. Springer, 2025.

Angel Kodituwakku, Clark Xu, Daniel Rogers, David K Ahn, and Errin W Fulp. Temporal aspects of cyber threat intelligence. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 6207–6211. IEEE, 2023.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Ortal Lavi, Ofir Manor, Tomer Schwartz, Andrés F Murillo, Ayoub Messous, Motoyoshi Sekiya, Junichi Suga, Kenji Hikichi, and Yuki Unno. Fine-tuning large language models for network traffic analysis in cyber security. In *2024 IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 45–50. IEEE, 2024.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

Xiaoqun Liu, Feiyang Yu, Xi Li, Guanhua Yan, Ping Yang, and Zhaohan Xi. Benchmarking llms in an embodied environment for blue team threat hunting. *arXiv preprint arXiv:2505.11901*, 2025.

Ariana Martino, Michael Iannelli, and Coleen Truong. Knowledge injection to counter large language model (llm) hallucination. In *European Semantic Web Conference*, pp. 182–185. Springer, 2023.

Vasileios Mavroeidis and Siri Bromander. Cyber threat intelligence model: an evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In *2017 European Intelligence and Security Informatics Conference (EISIC)*, pp. 91–98. IEEE, 2017.

Yuqiao Meng, Luoxi Tang, Dazheng Zhang, Rafael Brens, Elvys J Romero, Nancy Guo, Safa Elkefi, and Zhaohan Xi. Small agent group is the future of digital health. *arXiv preprint arXiv:2602.08013*, 2026.

Kevin Mitnick. *The art of invisibility: The world's most famous hacker teaches you how to be safe in the age of big brother and big data*. Little, Brown, 2017.

MITRE Corporation. Common attack pattern enumeration and classification (capec). `https://capec.mitre.org/`, 2024a.

MITRE Corporation. Common weakness enumeration (cwe). `https://cwe.mitre.org/`, 2024b.

Vrushket More, Lyra Lu, Zeyu Ding, Zhaohan Xi, Seth Mizia, and Nancy L Guo. Theramind: a multi-llm ensemble for accelerating drug repurposing in lung cancer via case report mining. *npj Precision Oncology*, 2026.

Taewoo Nam. Understanding the gap between perceived threats to and preparedness for cybersecurity. *Technology in society*, 58:101122, 2019.

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

Alexander Tobias Neumann, Yue Yin, Sulayman Sowe, Stefan Decker, and Matthias Jarke. An llm-driven chatbot in higher education for databases and information systems. *IEEE Transactions on Education*, 68(1):103–116, 2024.

Zhenxing Niu, Haodong Ren, Xinbo Gao, Gang Hua, and Rong Jin. Jailbreaking attack against multimodal large language model. *arXiv preprint arXiv:2402.02309*, 2024.

Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proceedings of the internet measurement conference*, pp. 478–485, 2019.

Alexis M Pitney, Spencer Penrod, Molly Foraker, and Suman Bhunia. A systematic review of 2021 microsoft exchange data breach exploiting multiple vulnerabilities. In *2022 7th international conference on smart and sustainable technologies (SpliTech)*, pp. 1–6. IEEE, 2022.

Nagaraju Pureti. Zero-day exploits: Understanding the most dangerous cyber threats. *International Journal of Advanced Engineering Technologies and Innovations*, 1(2):70–97, 2022.

Md Rayhanur Rahman, Rezvan Mahdavi-Hezaveh, and Laurie Williams. A literature review on mining cyberthreat intelligence from unstructured texts. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pp. 516–525. IEEE, 2020.

Shuo Ren, Can Xie, Pu Jian, Zhenjiang Ren, Chunlin Leng, and Jiajun Zhang. Towards scientific intelligence: A survey of llm-based scientific agents. *arXiv preprint arXiv:2503.24047*, 2025.

Maria Rigaki, Carlos Catania, and Sebastian Garcia. Hackphyr: A local fine-tuned llm agent for network security environments. *arXiv preprint arXiv:2409.11276*, 2024.

Nathan E Rosenblum, Xiaojin Zhu, Barton P Miller, and Karen Hunt. Learning to analyze binary computer code. In *AAAI*, pp. 798–804, 2008.

Sakari Ruotsalainen. Microsoft os and office patching procedure in an enterprise environment. 2018.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. Llm-sr: Scientific equation discovery via programming with large language models. *arXiv preprint arXiv:2404.18400*, 2024.

Adam Shostack. *Threat modeling: Designing for security*. John wiley & sons, 2014.

Leslie F Sikos. Defensive generative ai. In *Generative AI in Cybersecurity*, pp. 1–24. Springer, 2025.

Aditya K Sood and Richard J Enbody. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE security & privacy*, 11(1):54–61, 2012.

Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. Boms away! inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pp. 1–13, 2024.

Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 635–647, 2009.

Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. Mitre att&ck: Design and philosophy. 2018.

Tao Sun, Jian Xu, Yuanpeng Li, Zhao Yan, Ge Zhang, Lintao Xie, Lu Geng, Zheng Wang, Yueyan Chen, Qin Lin, et al. Bitsai-cr: Automated code review via llm in practice. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 274–285, 2025.

Luoxi Tang, Tharunya Sundar, Shuai Yang, Ankita Patra, Manohar Chippada, Giqi Zhao, Yi Li, Riteng Zhang, Tunan Zhao, Ting Yang, et al. Are llms ready for english standardized tests? a benchmarking and elicitation perspective. *arXiv preprint arXiv:2505.17056*, 2025.

Mevlut Serkan Tok and Baris Celiktas. Muddywater apt group and a methodology proposal for macro malware analysis. *Bilişim Teknolojileri Dergisi*, 12(3):253–263, 2019.

Bianca Trinkenreich, Fabio Calefato, Geir Hanssen, Kelly Blincoe, Marcos Kalinowski, Mauro Pezzè, Paolo Tell, and Margaret-Anne Storey. Get on the train or be left on the station: Using llms for software engineering research. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 1503–1507, 2025.

Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, and Andras Iklody. Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of the 2016 ACM on workshop on information sharing and collaborative security*, pp. 49–56, 2016.

Jiawei Wang, Renhe Jiang, Chuang Yang, Zengqing Wu, Makoto Onizuka, Ryosuke Shibasaki, Noboru Koshizuka, and Chuan Xiao. Large language models as urban residents: An llm agent framework for personal mobility generation. *Advances in Neural Information Processing Systems*, 37:124547–124574, 2024.

Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, et al. A comprehensive survey in llm (-agent) full stack safety: Data, training and deployment. *arXiv preprint arXiv:2504.15585*, 2025.

Tianyi Wang and Kam Pui Chow. Automatic tagging of cyber threat intelligence unstructured data using semantics extraction. In *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 197–199. IEEE, 2019.

Sajana Weerawardhena, Paul Kassianik, Blaine Nelson, Baturay Saglam, Anu Vellore, Aman Priyanshu, Supriti Vijay, Massimo Aufiero, Arthur Goldblatt, Fraser Burch, et al. Llama-3.1-foundationai-securityllm-8b-instruct technical report. *arXiv preprint arXiv:2508.01059*, 2025.

Yusuke Yamauchi, Taro Yano, and Masafumi Oyamada. An empirical study of llm-as-a-judge: How design choices impact evaluation reliability. *arXiv preprint arXiv:2506.13639*, 2025.

Zeliang Yu, Ming Wen, Xiaochen Guo, and Hai Jin. Maltracker: A fine-grained npm malware tracker copiloted by llm-enhanced dataset. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1759–1771, 2024.

Boyang Zhang, Yicong Tan, Yun Shen, Ahmed Salem, Michael Backes, Savvas Zannettou, and Yang Zhang. Breaking agents: Compromising autonomous llm agents through malfunction amplification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34952–34964, 2025.

Jie Zhang, Hui Wen, Liting Deng, Mingfeng Xin, Zhi Li, Lun Li, Hongsong Zhu, and Limin Sun. Hackmentor: Fine-tuning large language models for cybersecurity. In *2023 IEEE 22nd international conference on trust, security and privacy in computing and communications (trustCom)*, pp. 452–461. IEEE, 2023.

Yuan Zhou, Peng Zhang, Mengya Song, Alice Zheng, Yiwen Lu, Zhiheng Liu, Yong Chen, and Zhaohan Xi. Zodiac: A cardiologist-level llm framework for multi-agent diagnostics. *arXiv preprint arXiv:2410.02026*, 2024a.

Zhen Zhou, Ziyuan Gu, Xiaobo Qu, Pan Liu, Zhiyuan Liu, and Wenwu Yu. Urban mobility foundation model: A literature review and hierarchical perspective. *Transportation Research Part E: Logistics and Transportation Review*, 192:103795, 2024b.

Zhuotong Zhou, Yongzhuo Yang, Susheng Wu, Yiheng Huang, Bihuan Chen, and Xin Peng. Magneto: A step-wise approach to exploit vulnerabilities in dependent libraries via llm-empowered directed fuzzing. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1633–1644, 2024c.

APPENDIX CONTENTS

A AI Use Statement **16**

B Reproducibility Statement **16**

C Limitations **16**

D Complementary Details of CTI **16**

D.1 ❶ Contextualization 16

D.2 ❷ Attribution 17

D.3 ❸ Prediction 18

D.4 ❹ Mitigation 18

E Additional Details of CTI Experiment for §2 **19**

E.1 Used Benchmark in Evaluation 19

E.2 Used Real-World Databases and Platforms in Evaluation 19

E.3 Task Description 20

E.4 Data Statistics 22

E.5 Model Name and Version 22

E.6 Evaluation Prompt Structure 22

F Complementary Details for §3 **24**

F.1 Algorithm 24

F.2 Human Annotation Strategy and Taxonomy Robustness 24

F.3 Determining Failure Mode Types 28

G Complementary Analysis **30**

G.1 Ablation study 30

G.2 Varying LLM performance 30

G.3 Root cause analysis 31

G.4 Additional Analyses of Intertwined Failure Modes 44

H Complementing Details of Defensive Strategy **45**

H.1 A causal intervention design to validate and mitigate CTI Failure Modes 45

H.2 Fine-grained mitigation tailored to specific failure mode subtypes 47

A AI USE STATEMENT

Generative AI tools were used only to assist with language editing, including improving readability, grammar, concision, and presentation of the manuscript, as well as minor LaTeX formatting assistance. Generative AI tools were not used to generate synthetic datasets; develop theoretical models or conceptual frameworks; formulate mathematical claims; provide ingredients for, or assist in writing, mathematical proofs; propose or refine research hypotheses; design or provide feedback on research methodology or experiments; implement the proposed methods; translate research materials; clean or reformat datasets; support qualitative or thematic data analysis; or interpret experimental results. We have reviewed and revised all AI-assisted writing and formatting. The authors take full responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

B REPRODUCIBILITY STATEMENT

We provide the information and artifacts necessary to reproduce our experiments throughout the paper and supplementary materials. Section 2.2 describes the evaluation setting and datasets, while Section 3 specifies the proposed failure-mode categorization methodology. Appendix D provides detailed task definitions, dataset statistics, model names and versions, and evaluation prompts; Appendix E provides the categorization algorithms, annotation protocol, and taxonomy construction details; Appendix F contains complementary analyses and ablations; and Appendix G documents the defensive intervention and its complete evaluation. We additionally release the implementation and associated experimental artifacts through the anonymized repository at `https://anonymous.4open.science/r/CTI-Gap-3CBA`.

C LIMITATIONS

Our study focuses on representative CTI tasks, datasets, and LLMs that cover the four major stages considered in this work. While this provides broad coverage of current CTI practice, the threat landscape, model families, and cybersecurity tooling continue to evolve. Accordingly, the relative prevalence of individual failure-mode subtypes may shift as new models, threat sources, and operational settings emerge. Our taxonomy is intended to characterize recurring reasoning patterns rather than provide an exhaustive enumeration of every possible CTI failure, and future work can extend it as new threat behaviors and LLM-based CTI workflows arise.

D COMPLEMENTARY DETAILS OF CTI

This appendix section provides additional details on how different CTI stages are conducted with the involvement of various techniques, complementing Section §2.1.

D.1 ❶ CONTEXTUALIZATION

The contextualization stage transforms raw, fragmented observations into structured and actionable intelligence. Below we elaborate on the specific techniques commonly employed in real world CTI operations, with a natural sequence in practice.

- **Topic Modeling.** Topic modeling techniques such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003), Non-negative Matrix Factorization (Lee & Seung, 1999), and more recent neural topic models (Dieng et al., 2020) are applied to large corpora of unstructured CTI text, including threat reports, incident tickets, and log annotations. These models group documents or paragraphs into coherent themes, enabling analysts to identify clusters of related activity such as "phishing campaigns leveraging Office macros" or "ransomware families exploiting VPN vulnerabilities." In practice, topic modeling supports early triage by prioritizing threat feeds that share thematic overlap with active campaigns in a given sector.
- **Event Extraction.** Event extraction identifies structured incidents—*who did what, when, and how*—from unstructured logs or reports. Natural Language Processing (NLP) pipelines

detect entities such as vulnerabilities (CVEs), indicators of compromise (IP addresses, file hashes), and TTPs (MITRE ATT&CK techniques (Strom et al., 2018)), then associate them with temporal markers and actor actions. In Security Operations Centers (SOCs), event extraction enables long-form reports to be converted into structured JSON objects or STIX bundles, which can be automatically ingested into SIEM platforms such as Splunk or Elastic. This reduces manual parsing effort and ensures consistent representation across heterogeneous data sources.

- **Knowledge Base Mapping.** Knowledge base mapping aligns extracted entities and events with standardized taxonomies such as CVE (CVE Program, 2024), CWE (MITRE Corporation, 2024b), MITRE ATT&CK (Strom et al., 2018), and CAPEC (MITRE Corporation, 2024a). This process typically involves both exact matching (e.g., direct CVE ID resolution) and approximate entity linking (e.g., mapping the phrase "remote PowerShell execution" to ATT&CK T1059.001). In CTI practice, knowledge base mapping is essential for interoperability: intelligence can be shared across organizations using a common language of identifiers, enabling correlation of local incidents with global adversary behaviors.
- **Information Retrieval.** Information retrieval systems allow analysts to ground their findings in relevant historical reports and external databases. Implementations include keyword- and embedding-based retrieval from proprietary CTI feeds (e.g., Recorded Future, Mandiant Advantage), open-source repositories (e.g., AlienVault OTX, AbuseIPDB), and structured vulnerability catalogs such as NVD (National Institute of Standards and Technology (NIST), 2024) and the CISA KEV catalog (Cybersecurity and Infrastructure Security Agency (CISA), 2024). Retrieval pipelines often combine lightweight keyword filters for precision with dense embedding search for semantic coverage. Analysts use these systems to verify whether a newly observed domain has prior associations with known malware campaigns, or whether a vulnerability is actively being exploited.
- **Operational Integration.** In SOC environments, the above techniques are integrated into semi-automated pipelines. For example, a suspicious DNS query may trigger automated enrichment: (i) topic modeling identifies thematic overlap with phishing campaigns, (ii) event extraction links the query to specific malware families, (iii) knowledge base mapping ties the observed behavior to ATT&CK techniques, and (iv) information retrieval retrieves prior cases of infrastructure reuse. The result is a structured incident summary that equips analysts with actionable context for attribution, prediction, and mitigation.

D.2 ❷ ATTRIBUTION

Attribution in cyber threat intelligence involves identifying the adversary or campaign responsible for observed malicious activity. This process requires combining technical indicators with contextual and behavioral evidence. Below we describe the primary techniques used in real-world practice, along with how they are operationalized by analysts.

- **Named Entity Recognition (NER).** NER systems are used to extract structured entities such as malware families, infrastructure elements, or actor names from unstructured reports and incident logs. For example, extracting references to *APT28*, *Mimikatz*, or *QakBot* across heterogeneous feeds helps analysts consolidate threat narratives.
- **Relation and Event Extraction.** Beyond isolated entities, attribution requires uncovering relationships among adversary tactics, techniques, and infrastructure. Relation extraction techniques map, for instance, shared IP ranges between phishing campaigns or code reuse across malware variants, while event extraction captures sequences of actions such as exploitation followed by lateral movement (Husari et al., 2017).
- **Infrastructure Correlation.** Adversaries frequently reuse or repurpose command-and-control (C2) infrastructure. CTI teams apply graph-based correlation to link domain registrations, TLS certificates, and hosting providers, enabling attribution of otherwise fragmented observations to known actor toolkits or campaigns (Stone-Gross et al., 2009; Antonakakis et al., 2017).
- **Stylistic and Linguistic Profiling.** Analysts also consider linguistic cues in adversary communications or malware code artifacts. Stylometry and compilation fingerprinting

methods can identify patterns in variable naming, debugging strings, or grammar usage, which help to tie malware development back to particular groups (Rosenblum et al., 2008).

- **Campaign Graph Construction.** To synthesize diverse evidence, CTI analysts build structured graphs of adversarial campaigns, linking entities, infrastructure, and TTPs across incidents. Graph construction enables propagation of attribution hypotheses and detection of actor evolution over time.

D.3 ❸ PREDICTION

Prediction in CTI involves forecasting adversarial actions, exploitation likelihood, and campaign evolution. Unlike contextualization and attribution, prediction requires reasoning under temporal uncertainty and incomplete information. In practice, analysts and automated systems deploy a range of data-driven and model-based techniques to anticipate threats:

- **Historical correlation and trend analysis.** Analysts correlate prior incidents and intrusion campaigns to identify recurring attacker playbooks. For example, statistical methods on longitudinal CVE exploitation data help assess whether recently disclosed vulnerabilities follow exploitation trends of past families (Bilge & Dumitraş, 2012; Allodi, 2017).
- **Exploit prediction scoring.** Models such as the Exploit Prediction Scoring System (EPSS) estimate the probability that a vulnerability will be exploited within a given time window, using features such as vulnerability metadata, CVSS scores, and real-world exploit observations (Bullough et al., 2017).
- **Temporal modeling of campaign progression.** Recurrent neural networks and temporal point processes capture how campaigns unfold over time, modeling likely transitions from initial access to follow-on payloads such as ransomware.
- **Temporal forrcasting.** LLMs predict which TTPs an actor is likely to employ next or what impact. This supports proactive defense, such as generating detection rules for tactics not yet observed in the ongoing campaign (Sood & Enbody, 2012).
- **Threat propagation simulation (synthesis).** Agent-based and epidemic-style models simulate the spread of malware or worms across interconnected systems, forecasting infection curves and propagation likelihood (Cheng et al., 2010).

D.4 ❹ MITIGATION

Mitigation is the final stage of cyber threat intelligence (CTI), where enriched analysis and attribution results are translated into concrete defensive measures. Unlike contextualization or attribution, which primarily generate insights, mitigation requires actionable transformations that directly alter security posture. Below, we introduce the major classes of techniques commonly adopted in real-world CTI practice.

- **Detection rule generation.** Security teams design, validate, and deploy detection rules in languages such as Sigma and YARA to capture specific threat behaviors. In practice, detection rules are tuned iteratively: analysts translate threat reports into rule signatures, test them against telemetry or sandbox logs, and refine them to reduce false positives while ensuring coverage of attacker tradecraft.
- **Mitigation efficacy evaluation.** Beyond applying countermeasures, analysts must assess their effectiveness. Approaches include red-teaming exercises, breach-and-attack simulation (BAS) platforms, and adversary emulation scenarios that replay known TTPs to test whether mitigations succeed in preventing, detecting, or containing malicious activity (Shostack, 2014).
- **Response playbook recommendation.** Structured playbooks standardize incident response by encoding lessons from CTI. These include step-by-step containment and recovery actions tailored to adversary campaigns (e.g., disabling compromised accounts, isolating infected subnets). Orchestration tools such as SOAR platforms automate playbook execution, integrating CTI feeds into dynamic workflows (Mitnick, 2017).

- **Summarization.** Finally, mitigation intelligence is communicated through concise, contextualized reports for executives and IT operators. Summarization synthesizes prioritized vulnerabilities, mapped mitigations, and recommended workflows. This ensures decision makers understand trade-offs between operational impact and security gains, and supports cross-team coordination in enterprise-scale defense (Mavroeidis & Bromander, 2017).

E Additional Details of CTI Experiment for §2

E.1 Used Benchmark in Evaluation

Table 6 presents the original scales of the datasets used in our evaluation.

Table 6: Used cybersecurity benchmarks in our evaluations.

Benchmark	Focus	#Data	#Task	#Source
CTIBench (Alam et al., 2024)	Cyber Threat Intelligence	5,610	5	N/A
SevenLLM-Bench (Ji et al., 2024)	Report Analyzing	92,701	28	N/A
SWE-Bench (Jimenez et al., 2023)	Bug fixing	2,294	12	1
CYBERTEAM (Liu et al., 2025)	Blue-team threat hunting	452,293	30	23

E.2 Used Real-World Databases and Platforms in Evaluation

In addition to benchmarks, we incorporate several real-world cybersecurity databases and intelligence platforms to ensure that our evaluation settings reflect practical CTI usage. Each database provides complementary coverage across the CTI stages (§2.1), and we detail both their scope and our methodology for leveraging their information.

National Vulnerability Database (NVD). The NVD (National Institute of Standards and Technology (NIST), 2024) serves as the canonical repository for software vulnerabilities and their CVSS severity scores. We utilize NVD entries to support all ❶-❹ tasks. Specifically, we map vulnerabilities in benchmark items to NVD records in order to standardize CVE identifiers, extract official CVSS vector strings, and obtain temporal metadata (publication and modification dates). These fields allow us to align system-environment observations with ground-truth vulnerability characteristics, and to test forecasting models that predict exploit likelihood (e.g., by contrasting NVD base scores against EPSS-derived estimates).

Exploit Prediction Scoring System (EPSS). EPSS (Jacobs et al., 2021) provides probabilistic estimates of the likelihood that a given CVE will be exploited in the wild. We use EPSS scores directly for ❸ prediction tasks, both as a source of labels (ground-truth exploitation likelihood) and as a reference distribution to evaluate calibration of LLM-based forecasts. EPSS time-series updates also enable temporal correlation experiments, where we assess whether models capture shifts in exploitation likelihood following disclosure, patch release, or threat-actor reuse.

MITRE ATT&CK and CAPEC. The ATT&CK (Al-Sada et al., 2024) knowledge base encodes adversarial tactics, techniques, and procedures (TTPs) in a structured taxonomy. We map CTI tasks involving entity extraction, campaign attribution, and mitigation alignment to ATT&CK entries. For instance, when evaluating ❷ attribution, extracted TTPs from LLM outputs are compared against ATT&CK technique identifiers to assess correctness. We also leverage CAPEC (Common Attack Pattern Enumeration and Classification) (MITRE Corporation, 2024a) to validate abstract attack patterns referenced in benchmark items, particularly for mapping contextualized logs or IOCs to higher-level adversarial behaviors.

MISP (Malware Information Sharing Platform). MISP (Wagner et al., 2016) serves as a community-driven threat intelligence sharing platform, containing structured feeds of indicators of compromise (IOCs), malware samples, and infrastructure metadata. We use MISP to enrich contextualization tasks (❶) by grounding benchmark instances in realistic IOC–malware–actor relationships. For example, when a dataset item involves resolving a suspicious domain, we verify its presence in MISP feeds and align it to associated threat actors or malware families. This enrichment supports evaluation of LLMs' ability to normalize IOCs and link them to campaigns.

VirusTotal. VirusTotal (Peng et al., 2019) aggregates antivirus detections and malware analysis results across a large corpus of submitted files, domains, and URLs. We leverage VirusTotal reports for ❶ contextualization and ❷ attribution tasks. In contextualization, VirusTotal tags (e.g., malware family labels, sandbox behavior summaries) serve as auxiliary ground-truth for tasks like malware family mapping. In attribution, we analyze infrastructure overlap by checking whether related domains or IPs have been co-reported in VirusTotal samples, allowing us to validate LLM predictions about infrastructure reuse.

Open Threat Exchange (OTX). AlienVault's OTX (Cherqi et al., 2021) provides community-curated threat pulses (collections of IOCs associated with specific campaigns or malware). We use OTX primarily for ❷ attribution and ❸ prediction: pulses give us labeled groupings of IOCs tied to campaigns, which we then cross-check against LLM-predicted campaign attributions.

Security Advisories and Vendor Bulletins. Finally, vendor advisories (e.g., Microsoft, Cisco, Progress Software) and public CERT bulletins provide authoritative patching and mitigation recommendations. We incorporate these resources into ❹ mitigation tasks by aligning recommended countermeasures with benchmark items. For instance, in patch recommendation evaluation, the correct answer set is derived from vendor bulletins rather than from secondary threat reports. Similarly, YARA and Sigma rule examples are drawn from advisory-linked repositories, ensuring that response summarization tasks are grounded in practical remediation steps.

For consistency, we design a preprocessing pipeline that (i) normalizes identifiers (CVE, IOC, ATT&CK TTPs) across databases, (ii) aligns timeframes so that prediction tasks respect disclosure/exploitation chronology, and (iii) constructs ground-truth mappings between observations and actor/campaign/mitigation entities. We are thus able to systematically evaluate LLMs across all CTI stages using both controlled benchmarks (Table 1) and real-world ground-truth data.

E.3 Task Description

❶ Contextualization.

Affected Systems (F1). Binary decision per asset: is a listed host/application impacted by the described CVE/IOC set (yes/no). *Example:* decide whether `Exchange 2019 CU12` is affected given a CVE vector and server build.

Attack Infrastructure (F1). Binary decision per indicator: determine whether an IP/domain/URL belongs to adversary C2 or delivery infrastructure. *Example:* classify `cdn-upd[.]com` as campaign infra vs. benign CDN.

Vulnerability Linking (Acc). Multi-class assignment of correct CVE(s) from candidates based on logs/snippets. *Example:* map an IIS error pattern to {`CVE-2021-34473`} among distractors.

Malware Family Mapping (F1). Binary decision per candidate family: does observed behavior/artifacts match the family's signature (yes/no). *Example:* tag samples as belonging to a loader vs. banking trojan family.

IOC Normalization (F1). Binary correctness for canonicalizing raw IOCs (type+value) against gold forms. *Example:* normalize `hxxp://ex[.]ample[.]com/login` to `http://ex.ample.com/login` (URL).

Threat Report Alignment (BLEU). Text similarity between a generated one-sentence alignment/abstract and a reference summary of the most relevant report. *Example:* produce a synopsis that matches the gold advisory linkage.

Event Timeline Construction (BLEU). Compare generated event sequence text to a gold timeline. *Example:* "phish → beacon → lateral → exfil" vs. reference steps.

Graph Population (Acc). Multi-label slot filling for nodes/edges in an event graph (accuracy over required triples). *Example:* add {host–used_tool–T1059} and {user–compromised_via–phish} edges correctly.

Source Reliability Scoring (AUC). Binary scoring of source credibility (reliable vs. suspect) with probabilistic output; evaluated by ROC-AUC. *Example:* score a paste site vs. vendor advisory on the same IOC claim.

❷ Attribution.

Threat Actor Linking (Acc). Multi-class assignment of the most plausible actor profile(s) from candidates. *Example:* pick the actor whose historical TTP set matches observed techniques.

TTP Extraction (F1). Binary decision per candidate technique ID: is T#### evidenced (yes/no). *Example:* confirm `T1059` (command execution) from process tree snippets.

Campaign Attribution (Acc). Multi-class selection of a campaign label among candidates. *Example:* assign activity to a 2023 spearphishing campaign vs. a 2024 credential-harvest run.

Infrastructure Reuse (F1). Binary decision per linkage: does an IOC show reuse across events (yes/no). *Example:* mark `203.0.113.7` as reused across two clusters within 30 days.

Language/Style Profiling (Acc). Multi-class style attribution (e.g., build system, macro style, lure phrasing). *Example:* assign documents to a known lure/style family.

False Flag Detection (F1). Binary decision: is an observed signature intentionally misleading (yes/no). *Example:* detect planted strings mimicking a different actor's toolkit.

Evidence Weighting (BLEU). Textual rationale summarizing which evidence most supports the conclusion; compared to a gold rationale via BLEU. *Example:* generate a short justification prioritizing sandbox logs over OSINT.

Relation Graph Building (F1). Binary decision per candidate relation triple (entity–relation–entity). *Example:* validate {domain→hosts→IP} and reject spurious actor edges.

❸ Prediction.

Exploit Likelihood (AUC). Binary probability that a CVE will be exploited within horizon h (e.g., 30/90 days); evaluated with ROC-AUC. *Example:* score `CVE-YYYY-XXXX` as $p = 0.41$ for 30-day horizon.

Impact Forecast (BLEU). Generate a short impact summary (availability/integrity/confidentiality and severity band) and compare to a reference text. *Example:* "high integrity, medium availability; critical if unpatched."

Target Sector Prediction (Acc). Multi-class selection of likely sectors to be targeted. *Example:* choose {healthcare, finance} from a sector set.

Campaign Escalation (AUC). Binary probability that activity will escalate (e.g., hands-on-keyboard, ransomware) within h; measured by ROC-AUC. *Example:* output $p = 0.32$ escalation within 14 days.

❹ Mitigation.

Patch Recommendation (F1). Binary decision per candidate patch/hotfix: apply (yes/no) given product/version constraints. *Example:* select KB# for a specific Windows build; skip superseded fixes.

Rule Generation (YARA) (BLEU). Generate a detection rule text and compare to a canonical reference via BLEU. *Example:* produce a YARA rule body that matches gold strings/conditions.

Response Summarization (BLEU). Produce a concise remediation summary aligned to gold text. *Example:* "disable external OWA, apply patch KB…, add WAF rule …".

Mitigation–TTP Mapping (Acc). Multi-class mapping from observed TTPs to the correct mitigation set. *Example:* map {T1059, T1027} to script-blocking and DLL-search-order hardening.

Defensive Playbook Gen (BLEU). Generate stepwise response playbook text; similarity to reference measured by BLEU. *Example:* contain→eradicate→recover with host/network steps.

Countermeasure Ranking (NDCG). Rank candidate defenses by expected risk reduction; graded relevance compared to an ideal ranking. *Example:* prioritize patching and credential hygiene over low-yield blocks.

Incident Ticket Generation (Acc). Multi-class assignment for ticket fields (category, priority, assignment group). *Example:* classify as `malware/P2` with SOC-triage group.

Table 7: Per-task instance counts in our evaluations. Counts may overlap across tasks.

❶ Contextualization	
Affected Systems	3.9K
Attack Infrastructure	4.1K
Vulnerability Linking	3.5K
Malware Family Mapping	2.1K
IOC Normalization	3.1K
Threat Report Alignment	2.0K
Event Timeline Construction	1.5K
Graph Population	2.3K
Source Reliability Scoring	1.6K
❷ Attribution	
Threat Actor Linking	2.1K
TTP Extraction	3.0K
Campaign Attribution	1.8K
Infrastructure Reuse	1.2K
Language/Style Profiling	1.3K
False Flag Detection	0.9K
Evidence Weighting	1.5K
Relation Graph Building	1.8K

❸ Prediction	
Exploit Likelihood	2.5K
Impact Forecast	1.5K
Target Sector Prediction	1.3K
Campaign Escalation	1.1K
❹ Mitigation	
Patch Recommendation	1.9K
Rule Generation (YARA)	1.2K
Response Summarization	1.8K
Mitigation-TTP Mapping	1.5K
Defensive Playbook Gen	1.1K
Countermeasure Ranking	1.3K
Incident Ticket Generation	1.2K

E.4 Data Statistics

Table 7 lists the number of instances we collected and used in evaluations.

E.5 Model Name and Version

Here we detail the used LLMs in our extensive evaluation (§2.2), corresponding to the abbreviated names used in Table 2 and in Section §4: G5: GPT-5, CLD: Claude Sonnet 4, GEM: Gemini 2.5, LL70: Llama-3.1-70B-Instruct, MIX: Mixtral-8x7B-Instruct-v0.1, QWN: Qwen2.5-14B-Instruct, FSC: Foundation-Sec-8B (Weerawardhena et al., 2025), ZYS: ZySec-AI-SecurityLLM (Sikos, 2025), LLY: Lily-Cybersecurity-7B-v0.2 (Briliyant et al., 2025), CBS: CyberBase-13b (CyberNative, 2024), SPT: clouditera-secgpt (clouditera, 2025), DHT: DeepHat-V1-7B (DeepHat, 2025).

E.6 Evaluation Prompt Structure

Below we provide our evaluation prompt used in §2.2:

Role. You are a cybersecurity threat-intelligence (CTI) analyst assistant *and* strict schema-enforcer. Convert only the provided inputs (docs, logs, IOCs, CVEs, ATT&CK, advisories) into the JSON contract below—*no prose*. Ground every field in supplied evidence; never invent identifiers. Honor `<SNAPSHOT_DATE>` as a hard knowledge freeze. Normalize and deduplicate CTI entities (CVE, ATT&CK, actor, IOC). If evidence is insufficient, return `"status":"NEED_MORE_CONTEXT"` with `missing_fields`. Respect safety (no exploit guidance) and determinism (temperature=`<TEMP>`, top_p=`<TOPP>`). Return exactly one JSON object and nothing else.
Objective. Solve `<TASK_NAME>` within `<CTI_STAGE>` (`CONTEXTUALIZATION | ATTRIBUTION | PREDICTION | MITIGATION`) using only the provided inputs at/before `<SNAPSHOT_DATE>`.
Inputs

- Case ID: `<CASE_ID>`
- Snapshot date (ISO): `<SNAPSHOT_DATE>`
- Source docs (IDs + short snippets): `<DOC_LIST>`
- Structured feeds: `<STRUCT_FEEDS>`
- Task guidance: `<TASK_GUIDANCE>`

- Output profile (choose fields to populate): <OUTPUT_PROFILE>

.....

Operating Rules

- Use only provided inputs. No external browsing or unstated facts.
- Do not fabricate CVE/TTP/actor names. Use exact IDs when given.
- If critical evidence is missing, return "status":"NEED_MORE_CONTEXT" and list missing_fields.
- Safety: no exploit code or offensive guidance. Mitigation only.
- Determinism: temperature=<TEMP>, top_p=<TOPP>.

.....

Output Contract (return *one* JSON only)

```
{
  "status": "OK | NEED_MORE_CONTEXT | UNSUPPORTED",
  "task": "<TASK_NAME>",
  "case_id": "<CASE_ID>",
  "snapshot_date": "<YYYY-MM-DD>",
  "answer": {
    "ioc_normalization": [{
      "raw":"<str>","type":"ipv4|domain|url|hash","value":"<canon>","
    first_seen":"<date?>",
      "tags":["<malware?>","<actor?>"]
    }],
    "vuln_linking": {
      "cve_candidates":[{"cve_id":"CVE-YYYY-XXXX","score":0.0-1.0}],
      "vector_string":"<CVSS3/4?>"
    },
    "malware_mapping": {
      "family_candidates":[{"name":"<family>","score":0.0-1.0}],
      "aliases":["<aka?>"], "capabilities":["<tags>"]
    },
    "event_timeline": [{
      "t":"<ISO>","type":"<beacon|phish|lateral|exfil>",
      "artifacts":["<IOC|host|user>"], "source_ref":"<DOC_ID>"
    }],
    "actor_linking": {
      "actor_candidates":[{"name":"<actor>","score":0.0-1.0}],
      "shared_ttps":["T####"], "infra_overlap":[{"indicator":"<ip|
    domain>","match":"exact|fuzzy"}]
    },
    "ttp_extraction": [{"technique_id":"T####","sub":"T####.###?","
    evidence_ref":"<DOC_ID>"}],
    "campaign_attribution": {"name":"<label>","score":0.0-1.0,"
    rationale_tags":["<sector?>","<geo?>"]},
    "false_flag": {"likelihood":0.0-1.0,"signals_for":["<s>"],"
    signals_against":["<s>"]},
    "exploit_likelihood": {"cve_id":"CVE-YYYY-XXXX","horizon_days
    ":<7|30|90>,
                           "prob_exploit":0.0-1.0,"drivers":["<poc
    ?>","<reuse?>"]},
    "impact_forecast": {"impact_vector":["<A|I|C>"],"severity_band":"
    low|med|high|critical","uncertainty":0.0-1.0},
    "target_sector": [{"name":"<NAICS-like>","prob":0.0-1.0}],
    "escalation": {"prob":0.0-1.0,"signals":["<toolchain shift>","<
    tempo>"]},
    "patch_recommendation": {"affected_assets":["<product|version>"],
                             "patches":[{"kb_or_id":"<vendor-ID>","
    priority":"P1|P2|P3"}],
                             "prechecks":["<backup?>","<downtime?>"]},
    "rule_generation": {"rule_type":"YARA|Sigma","rule_name":"<name
    >","rule_body":"<escaped>",
```

```
                        "test_iocs":["<ioc1>","<ioc2>"]},
    "countermeasure_ranking": [{"mitigation_id":"<ATT&CK M###|vendor
    >","title":"<short>",
                                "effort":"low|med|high","expected_gain
    ":"<short>"}],
    "incident_ticket": {"category":"<phishing|malware|ransomware
    |...>",
                        "priority":"P1|P2|P3","work_notes":["<steps
    >"],
                        "required_artifacts":["<pcap?>","<edr?>"]}
  },
  "confidence": 0.0-1.0,
  "justification": "<=40 words, terse, evidence-based>",
  "evidence_refs": ["<DOC_ID or IOC or CVE>", "..."],
  "metadata": {
    "stage": "<CONTEXTUALIZATION|ATTRIBUTION|PREDICTION|MITIGATION>",
    "assumptions": ["<short>"],
    "missing_fields": ["<if status=NEED_MORE_CONTEXT>"]
  }
}
```

Scoring & Tie-Breaks

- Prefer precise IDs (CVE, ATT&CK T#, actor handles) and multi-source corroboration.
- Resolve conflicts by source quality, recency ($\leq$ <SNAPSHOT_DATE>), and internal consistency.

Run Settings (fill before inference)

- <TASK_NAME> = <...> <CTI_STAGE> = <...>
- <CASE_ID> = <...> <SNAPSHOT_DATE> = <YYYY-MM-DD>
- <DOC_LIST> = <[ID:desc, ...]> <STRUCT_FEEDS> = <...>
- <TASK_GUIDANCE> = <...> <OUTPUT_PROFILE> = <...>
- <TEMP> = <0.0-0.3> <TOPP> = <0.8-1.0>

F Complementary Details for §3

F.1 Algorithm

This section presents the algorithms (Algo. 1, 2, and 3) complementing §3.

F.2 Human Annotation Strategy and Taxonomy Robustness

Our methodology involves human annotation at three distinct stages: (i) manually labeling anchor instances during stratification (RQ_1), (ii) initializing and refining the failure-mode taxonomy (RQ_2), and (iii) resolving uncertain assignments identified by the multi-agent procedure (RQ_3). We therefore establish a unified annotation method to ensure that human decisions are consistent across these stages. Importantly, the role of human annotators differs across the pipeline. During RQ_1 and RQ_2, human annotation contributes to identifying failures and constructing the taxonomy, whereas during RQ_3, the taxonomy has already been stabilized and human effort is used only to adjudicate ambiguous instance-level assignments.

Annotators and expertise. The annotation was conducted by **7 annotators**, including **4 graduate researchers** with research experience in cybersecurity and **3 professional cybersecurity researchers**. All annotators had prior experience with cyber threat intelligence or closely related security tasks, including vulnerability analysis, malware and threat analysis, MITRE ATT&CK, security operations, and LLM-based cybersecurity research. Their relevant research or professional experience ranged from **2 to 11 years**. The same annotator pool was used throughout the human-involved stages of the pipeline to avoid introducing systematic differences in interpretation between taxonomy construction and final verification.

Algorithm 1: Stratified Failure Sampling (RQ_1)

Input: Dataset $\mathcal{D} = \{x_i\}$, Models $\{y_i\}$, Refs $\{r_i\}$, Similarity measure $Sim(\cdot)$
Output: Failure Set $\mathcal{F}$

// 1. Score & Stratify
Compute scores $\mathcal{S} = \{s_i \mid s_i = Sim(y_i, r_i), \forall i\}$
Sort $\mathcal{S}$ and partition into quantile strata $\mathcal{Q} = \{Q_1, \ldots, Q_M\}$ (bin size $\delta = 0.05$)
// 2. Iterative Inspection
Initialize $\mathcal{F} \leftarrow \emptyset$, $Converged \leftarrow$ False
foreach *stratum* $Q_k \in \mathcal{Q}$ ***while*** $\neg Converged$ **do**
    Sample anchors $A_k \subset Q_k$ for manual labeling
    Derive score ranges R_{fail}, R_{pass} from labeled anchors A_k
    // Propagate labels to remaining x_i
    foreach $x_i \in Q_k \setminus A_k$ **do**
        if $s_i \in R_{fail}$ **then**
            $\mathcal{F} \leftarrow \mathcal{F} \cup \{x_i\}$
        else if $s_i \in R_{pass}$ **then**
            continue
        else
            Manual check x_i; **if** failed $\mathcal{F} \leftarrow \mathcal{F} \cup \{x_i\}$ // Boundary
        end
    end
    // 3. Termination Check
    Update failure instance distribution D_k
    if $D_k \approx D_{k-1}$ ***and*** *no new failure set found* **then**
        $Converged \leftarrow$ True
    end
end
return $\mathcal{F}$

Algorithm 2: Autoregressive Failure Mode Determination (RQ_2)

Input: Failure instances $\mathcal{D} = \{x_i\}_{i=1}^{N}$ (from stratification)
Human annotators H (for seeding and refinement)
Large Language Model f_θ (for assisted classification)
Stability threshold ε; coverage threshold ρ (e.g., 0.6)
Output: Stabilized taxonomy of failure modes $\mathcal{T}^*$
// Step 1: Initialization
Human annotators H inspect a subset $\mathcal{D}_0 \subset \mathcal{D}$;
Derive initial taxonomy $\mathcal{T}_0 = \{t_1, \ldots, t_k\}$;
// Step 2: LLM classification
foreach $x_j \in \mathcal{D} \setminus \mathcal{D}_0$ **do**
    Assign label $y_j \in \mathcal{T}_m \cup \{\texttt{other}\}$ using f_θ;
end
// Step 3: Refinement
Collect $\mathcal{O}_m = \{x_j \mid y_j = \texttt{other}\}$;
Human annotators H inspect $\mathcal{O}_m$;
If new modes $\Delta\mathcal{T}$ found, update taxonomy:
$\mathcal{T}_{m+1} \leftarrow \mathcal{T}_m \cup \Delta\mathcal{T}$;
// Repeat until convergence
Repeat Steps 2-3 until $\Delta\mathcal{T} = \varnothing$ and coverage $\geq \rho$;
Set $\mathcal{T}^* = \lim_{m\to\infty} \mathcal{T}_m$;
return stabilized taxonomy $\mathcal{T}^*$;

Annotation guideline. Before the main annotation, we developed a written annotation guideline that operationalizes the definitions of the failure modes used in this work. For each subtype, the guideline specifies: (1) the definition of the failure mode; (2) inclusion criteria indicating the evidence required for assigning the label; (3) exclusion criteria distinguishing the subtype from closely related failure modes; (4) representative positive and negative examples; and (5) common boundary cases

Algorithm 3: Human-in-the-loop Categorization of Failure Instances (RQ_3)

Input:
Failure instances $\mathcal{D} = \{x_i\}_{i=1}^{N}$ (from stratification)
Model set $\Theta = \{\texttt{GPT-5}, \texttt{Llama-4}, \texttt{Gemini}, \texttt{Claude}\}$
Human annotators H
Output:
Final categorized instances with reliable failure modes
`// Step 1: Multi-agent deliberation (round 1)`
foreach $x_i \in \mathcal{D}$ **do**
  Each model $a \in \Theta$ independently assigns $\hat{y}_i^{(1,a)} \in \mathcal{T}$;
end
`// Step 2: Repetition for stability (round 2)`
foreach $x_i \in \mathcal{D}$ **do**
  Each model $a \in \Theta$ observes $\{\hat{y}_i^{(1,b)}\}_{b \in \Theta}$;
  Refine reasoning and output $\hat{y}_i^{(2,a)}$;
end
`// Step 3: Human verification of uncertain cases`
Define uncertain set:
$\mathcal{U} = \{x_i \mid \exists a : \mathrm{Var}(\{\hat{y}_i^{(a)}\}_{a \in \Theta}) > 0 \ \textbf{or} \ \hat{y}_i^{(1,a)} \neq \hat{y}_i^{(2,a)}\}$;
Human annotators H inspect all $x_i \in \mathcal{U}$ and assign final labels;
`// Finalize results`
Instances $\mathcal{D} \setminus \mathcal{U}$ take majority-agreed labels from models;
Instances $\mathcal{U}$ take human-verified labels;
return categorized dataset with reliable failure mode assignments;

and the corresponding decision rules. Detailed subtype-specific criteria and examples are provided in Appendix F.3.

For each instance, annotators were provided with the original CTI task input, the corresponding model output, and the available reference evidence, such as benchmark answers, authoritative advisories, or structured CTI records. Annotators were instructed to assign a failure mode only when the observable discrepancy between the model output and the available evidence satisfied the corresponding decision criteria. Plausible explanations about why a model might have failed were not sufficient by themselves to justify a label.

Because different failure modes may coexist within the same CTI instance, annotators were allowed to assign **multiple failure modes** when each label was independently supported by the evidence. Thus, for an instance x_i, the human annotation is represented as a label set $Y_i \subseteq \mathcal{T}^*$ rather than forcing each instance into a single mutually exclusive category. Annotators could additionally mark an instance as `uncertain` when the available evidence was insufficient to distinguish between candidate failure modes.

Calibration and independent annotation. Before formal annotation, all annotators independently labeled a calibration set of **45 instances** sampled across the four CTI stages and the three high-level failure categories. The annotators then discussed disagreements in the calibration set to clarify ambiguous definitions and refine the annotation guideline. The guideline was frozen before the formal reliability evaluation, and calibration instances were excluded from the reported inter-annotator agreement statistics.

For the formal agreement study, we randomly sampled **320 instances**, stratified across CTI stages and failure-mode categories. Each instance was independently annotated by **3 annotators**. Annotators did not have access to the other annotators' decisions during this stage. Their original independent labels were preserved for the agreement analysis before any disagreement resolution was performed.

Inter-annotator agreement. Since the annotation is potentially multi-label, we evaluate inter-annotator agreement at both the individual-subtype and complete label-set levels. First, each failure subtype is treated as a binary presence/absence decision for every instance. We compute Krippendorff's α for each subtype and report the macro-average across all 15 subtypes. This formulation

Table 8: Inter-annotator agreement for human failure-mode annotation. Krippendorff's α is computed from binary presence/absence decisions for each subtype.

Failure subtype	**Krippendorff's α**
1.1 Co-mention bias	0.73
1.2 Exploitation bias	0.81
1.3 Confounding factors	0.62
1.4 Skewed source	0.69
1.5 Hierarchical metadata	0.78
2.1 Temporal contradiction	0.80
2.2 Conflicting reports	0.71
2.3 Semantic conflict	0.64
2.4 Divergent data structures	0.76
2.5 Misaligned standards	0.79
2.6 Counteracting generation	0.66
3.1 Distributional bias	0.74
3.2 Unseen patterns	0.68
3.3 Overfitted reasoning	0.63
3.4 Environmental unawareness	0.75
Macro average	**0.72**

measures whether annotators consistently determine the presence of each failure mode while accounting for agreement that may occur by chance.

We additionally measure agreement over the complete multi-label assignment using pairwise Jaccard similarity. For two annotators a and b, their agreement on instance x_i is

$$J_i^{(a,b)} = \frac{|Y_i^{(a)} \cap Y_i^{(b)}|}{|Y_i^{(a)} \cup Y_i^{(b)}|}. \tag{1}$$

We report the mean pairwise Jaccard similarity across instances and annotator pairs, together with the percentage of instances for which all annotators produced exactly the same label set. Across the independently annotated sample, we obtain a macro-average Krippendorff's α of **0.72**, a mean pairwise Jaccard agreement of **0.67**, and an exact label-set agreement of **58%**. Per-subtype agreement scores are reported in Table 8.

Disagreement resolution. After the agreement statistics were computed from the independent annotations, disagreement cases were adjudicated separately. Annotators first revisited the corresponding reference evidence and the written inclusion/exclusion criteria and discussed the source of the disagreement. A label was retained only when the available evidence satisfied its predefined annotation criteria. Cases that remained unresolved after discussion were reviewed by **a senior annotator**, with **13 years** of cybersecurity research or professional experience, whose decision served as the final adjudication. The resulting adjudicated labels were used as the final human annotations in subsequent analyses. Importantly, consensus labels obtained during adjudication were not used to compute inter-annotator agreement.

Selective human verification in $\mathbf{RQ}_3$. After the taxonomy $\mathcal{T}^*$ has stabilized in RQ_2, Algorithm 3 uses human inspection as a selective adjudication mechanism rather than as a replacement for automated categorization. In the first round, each model in the multi-agent set independently predicts the failure-mode assignment for an instance. During the second round, each model observes the first-round predictions of the other agents and re-evaluates the same instance.

We use two complementary signals to identify uncertain decisions. *Inter-model disagreement* captures cases in which heterogeneous LLMs assign different failure modes to the same evidence, whereas *cross-round instability* captures cases in which an individual model changes its assignment after observing the other agents' predictions. An instance is included in the uncertain set $\mathcal{U}$ whenever either type of instability is observed.

Human annotators inspect all instances in $\mathcal{U}$ using the same annotation guideline and evidence requirements described above. At this stage, annotators do not introduce new failure-mode categories:

RQ_3 operates exclusively over the stabilized taxonomy $\mathcal{T}^*$. Instances outside $\mathcal{U}$ retain the consensus assignment produced by the multi-agent procedure, whereas instances inside $\mathcal{U}$ receive the human-adjudicated assignment. This design separates *taxonomy discovery* from *instance-level verification* and concentrates human effort on cases for which automated decisions are demonstrably unstable. Empirically, fewer than 1.8% of instances require human verification during this stage.

Robustness of taxonomy discovery. A separate concern is whether the taxonomy discovered in RQ_2 is dependent on particular samples selected during initialization or stratification. Inter-annotator agreement alone cannot answer this question: high agreement indicates that annotators can apply a given taxonomy consistently, but does not demonstrate that the same taxonomy would be discovered under different sampling choices. We therefore perform a taxonomy-stability analysis by repeating the RQ_1-RQ_2 discovery procedure under multiple perturbations while keeping the underlying evaluation corpus and annotation guideline fixed.

We consider three sources of variation. First, we vary the random seed used to construct the initial human-inspected subset $\mathcal{D}_0$ in RQ_2, thereby testing sensitivity to **taxonomy initialization**. Second, within every score stratum Q_k in RQ_1, we independently resample the manually inspected anchor set $\mathcal{A}_k$, testing whether different **anchor selections** lead to different failure boundaries. Third, we vary the quantile width used for stratification around the default 5% setting, using **{2.5%, 5%, 10%}** quantile strata. For each perturbation condition, we repeat the complete discovery process using **5 independent runs**.

Measuring taxonomy stability. Because independently discovered taxonomies may use slightly different names for semantically similar categories, we do not compare taxonomy labels by lexical identity alone. Instead, all alternative taxonomies are evaluated on a fixed audit set $\mathcal{A}$ of **180 instances**. For each discovered subtype t, we obtain the set $S_t \subseteq \mathcal{A}$ of audit instances assigned to that subtype. Categories across two runs are then aligned using maximum-weight bipartite matching, where the matching weight between two subtypes is the Jaccard overlap of their assigned instance sets:

$$J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}. \tag{2}$$

This procedure allows semantically equivalent categories to be matched even when their surface names differ, while exposing genuinely split, merged, missing, or newly introduced categories.

We report four complementary stability measures: (1) **taxonomy recovery rate**, defined as the fraction of reference subtypes recovered in an alternative run; (2) **matched-category overlap**, measured as the mean Jaccard similarity between matched subtype instance sets; (3) **instance-level assignment stability**, measured as the mean Jaccard similarity between the complete multi-label assignments of the same audit instance across runs; and (4) **distribution stability**, measured by the Jensen-Shannon divergence (JSD) between the subtype prevalence distributions produced by different runs.

Across changes in initialization seeds, anchor samples, and stratification granularity, the resulting taxonomies recover **14/15** failure subtypes on average, with a taxonomy recovery rate of **0.91**, matched-category Jaccard similarity of **0.83**, and instance-level assignment Jaccard similarity of **0.79**. The average JSD between subtype distributions is **0.041**. These results indicate that the taxonomy is **highly** stable to the sampling choices involved in its construction rather than being an artifact of a particular initialization or anchor selection.

F.3 Determining Failure Mode Types

To systematically determine failure mode types, we compare model-generated outputs with ground-truth advisories or reference materials. Each type requires distinct criteria to establish whether a case constitutes a failure. Below, we provide detailed guidance for all failure modes.

1.1 Co-mention bias from raw threat incident. This type arises when the model assumes two entities are related simply because they appear together in the input. To detect it, we check whether the reference advisory explicitly states a relationship. If not, and the model still reports an association (e.g., linking a domain and a malware family that are merely co-mentioned in the same log file), we classify the case as co-mention bias. For instance, if a proxy log shows both "malware.exe" and

"example.com" but the advisory only validates the domain as malicious, any model statement that marks the executable as directly tied to the domain would be considered a co-mention failure.

1.2 Exploitation bias from deliberately reused IoCs. This failure occurs when the model treats historical IoCs as valid for a new incident without reference confirmation. To identify it, we compare whether the reference differentiates between "legacy" IoCs and those active in the reported exploitation. If the model does not honor this distinction, it is flagged. For example, if the advisory specifies that old C2 servers from 2021 were no longer used, but the model still lists them as indicators of the current 2024 campaign, we mark it as exploitation bias.

1.3 Confounding factors that correlate entities. This error stems from mistaking correlation for causation. To check, we review whether the model claims causal links between entities that the references treat only as co-occurring or related by context. For instance, if two APT groups both use the same loader malware, but references emphasize they are separate actors, any model conclusion that "Group A conducted the intrusion because the loader was observed" is a confounding-factor failure.

1.4 Skewed source. We detect this when the model bases its judgment on incomplete or biased evidence. To assess, we compare the diversity of sources reflected in the output against references that aggregate multiple reports. If the model reflects only one vendor's outdated claim while ignoring corrections from other advisories, it is considered a skewed-source failure. For example, if Symantec updates its report that the threat vector was RDP rather than phishing, but the model still reproduces the outdated phishing claim, the case is flagged.

1.5 Hierarchical metadata from attack chains. Here the failure lies in mishandling hierarchical structures like ATT&CK techniques and sub-techniques. We check whether the model's reported granularity matches the reference hierarchy. If a reference states "T1059.001: PowerShell execution" but the model collapses this to a generic "execution tactic," it shows a hierarchy error. Similarly, if the model treats a campaign label as equivalent to a single technique, the hierarchy is broken.

2.1 Temporal contradiction. This type occurs when the model confuses timelines. To identify it, we compare the time anchors in the model output (e.g., attack start date, patch release) with those in the references. If the model asserts events happened earlier or later than documented, it is a temporal contradiction. For example, if the advisory states that exploitation began in June 2024 but the model outputs "first exploited in 2022," we classify it as temporal contradiction.

2.2 Conflicting reports of attack contexts. This failure arises when multiple sources disagree and the model selects or merges them incorrectly. To determine it, we check whether the model reflects the final reconciled context in references. For example, if initial reports said "phishing email" but were later corrected to "supply-chain compromise," and the model insists on phishing without acknowledging the update, it is labeled as conflicting-context failure.

2.3 Semantic conflict. This error occurs when the model misinterprets terms or security concepts. To identify it, we verify whether the technical meaning in the reference aligns with the model's description. If references mention "privilege escalation" but the model interprets it as "initial access," the semantic mismatch makes it a semantic conflict.

2.4 Divergent data structures. This happens when the model fails to follow structured taxonomies. To evaluate it, we cross-check ATT&CK IDs, CVE structures, or CVSS vectors in the output with those in the references. If the model generalizes or drops detail (e.g., returning "execution" instead of "T1059.001"), the failure is classified as divergent data structure.

2.5 Misaligned knowledge and standards. This type emerges when the model applies outdated or inconsistent standards. We check whether the model's labels follow the same version of taxonomy as the references. For instance, if a CVE is officially scored with CVSS v3.1 but the model outputs a CVSS v2 vector, this is marked as misaligned knowledge.

2.6 Counteracting CTI generation and LLM alignment. This failure reflects cases where safety alignment suppresses accurate CTI reporting. To detect it, we compare whether the advisory confirms sensitive facts (e.g., that a zero-day is under active exploitation). If the model avoids mentioning it with a vague refusal (e.g., "details omitted for safety"), we classify it as alignment counteraction.

3.1 Distributional bias. We check if the model over-generalizes to common patterns. References may specify rare attack types, but if the model predicts frequent ones regardless, it is a distributional bias

Table 9: Ablation study by removing components from proposed failure mode categorization method. Performance closely matching human annotation demonstrates a near-human capability for failure mode discovery. Smaller gray values in parentheses denote standard deviations.

Method	Spurious Correlation					Contradictory Knowledge						Constrained Generalization			
	1.1 Co-men. bias	**1.2** Exploit. bias	**1.3** Confond. factor	**1.4** Skewed source	**1.5** Hier. metadata	**2.1** Temporal contra.	**2.2** Conflict report	**2.3** Semantic conflict	**2.4** Diver. data	**2.5** Misal. knowl.	**2.6** Counter. CTI	**3.1** Distribut. bias	**3.2** Unseen pattern	**3.3** Overfit. reasoning	**3.4** Env. unaware
Human annotation	4.14 (0.20)	1.71 (0.11)	15.72 (3.04)	6.16 (0.81)	2.05 (0.08)	13.98 (1.31)	22.29 (1.14)	4.14 (0.86)	4.53 (0.59)	1.97 (0.19)	7.45 (0.80)	21.79 (2.29)	23.13 (4.30)	7.64 (1.09)	11.76 (2.54)
w/o stratification	12.62 (3.24)	5.14 (0.47)	10.37 (2.69)	11.23 (2.86)	5.89 (0.58)	17.45 (2.63)	16.12 (2.63)	9.87 (1.03)	8.45 (2.36)	5.23 (0.43)	12.56 (1.62)	17.89 (0.97)	16.34 (1.05)	14.56 (1.86)	15.23 (0.33)
w/o human initiation	4.93 (0.65)	2.56 (0.24)	14.23 (0.74)	7.82 (1.52)	3.12 (0.71)	12.45 (2.25)	20.56 (1.47)	5.45 (0.28)	5.78 (1.58)	2.89 (0.39)	9.12 (1.40)	20.34 (3.74)	21.67 (2.43)	11.45 (1.06)	12.34 (2.19)
w/o multi-agent decision	2.34 (0.09)	6.78 (1.56)	21.56 (4.97)	3.45 (0.37)	0.56 (0.13)	9.12 (0.53)	28.45 (5.04)	1.67 (0.40)	2.89 (0.06)	0.92 (0.06)	4.12 (0.57)	30.23 (7.68)	13.45 (0.41)	5.67 (0.83)	6.78 (0.47)
w/o repetition	9.45 (1.43)	6.23 (0.42)	11.67 (0.86)	10.12 (2.07)	6.34 (0.69)	19.56 (4.89)	17.89 (1.81)	9.12 (0.65)	7.56 (1.42)	6.12 (0.52)	13.45 (3.38)	19.67 (3.43)	17.23 (1.46)	15.45 (0.70)	14.56 (4.07)
w/o human verification	3.67 (0.67)	2.89 (0.85)	16.45 (2.85)	5.45 (1.06)	2.67 (0.06)	14.89 (0.41)	23.67 (1.61)	4.89 (1.24)	3.78 (0.51)	2.56 (0.47)	6.45 (1.09)	23.12 (2.80)	24.56 (1.37)	9.34 (1.48)	10.23 (0.63)
Ours	4.12 (0.02)	1.87 (0.28)	15.50 (0.48)	6.31 (0.95)	1.98 (0.07)	13.86 (0.58)	22.33 (1.26)	3.96 (0.76)	4.49 (1.37)	1.85 (0.09)	7.77 (0.87)	23.34 (1.25)	25.06 (2.35)	10.02 (1.17)	10.80 (0.89)

failure. For example, when the advisory confirms a banking trojan but the model reports "ransomware" (because ransomware dominates training data), we flag it.

3.2 Unseen pattern from emerging threats. This failure occurs when the reference describes a novel exploitation unseen in prior data, and the model defaults to outdated templates. We identify it when the model misses new threat mechanics (e.g., cloud API abuse) and instead describes traditional server exploits. Such mismatches are categorized as unseen-pattern failures.

3.3 Overfitted reasoning. This type occurs when the model rigidly applies a reasoning shortcut. We check whether the model attributes intrusions repeatedly to the same group or vector, even when references show otherwise. For instance, if the reference states multiple groups use a given malware but the model always assigns it to a high-frequency CVE or APT, we label the case overfitted reasoning.

3.4 Environmental unawareness. This failure arises when the model ignores environmental scope. To check, we compare the affected platforms or industries in the output with those in the references. If the advisory states that only Linux servers are vulnerable but the model generalizes to "all enterprise systems," it is considered environmental unawareness.

G Complementary Analysis

G.1 Ablation study

Results. The ablation study in Table 9 shows a clear dependency across components. When we remove core architectural modules, especially multi-agent debate and stratification, the predicted subtype distribution departs sharply from the human gold standard, with unstable spikes emerging in categories such as *Confounding Factors (1.3)* and *Overfitted Reasoning (3.3)*. By comparison, disabling the human-in-the-loop steps (human initiation and verification) causes a smaller but still noticeable drop: the distribution drifts away from the ground truth, yet it remains far more stable than in settings where the automated reasoning machinery is removed.

Explanations and efficiency analysis. These results highlight a primary strength of our framework: **it cuts manual workload substantially without sacrificing efficacy.** Notably, even the w/o human verification variant still outperforms fully automated baselines (e.g., Clustering and Topic Modeling), indicating that the multi-agent architecture is doing most of the cognitive heavy lifting for failure categorization. In practice, this reframes the analyst's role from labor-intensive labeling to lightweight oversight. As a result, CTI teams can approach near-human precision while spending only a fraction of the time typically required for manual classification.

G.2 Varying LLM performance

Results and explanations. We further study the varying performance of different models. As shown in Figure 3, and consistent with the distributions across failure modes in Figure 1, our analysis reveals clear differences in how various LLMs handle distinct categories of errors: For failure modes that directly impair LLM inference (e.g., co-mentioned but irrelevant mitigation strategies or outdated metadata that reduce retrieval effectiveness), general-purpose LLMs tend to accumulate a higher

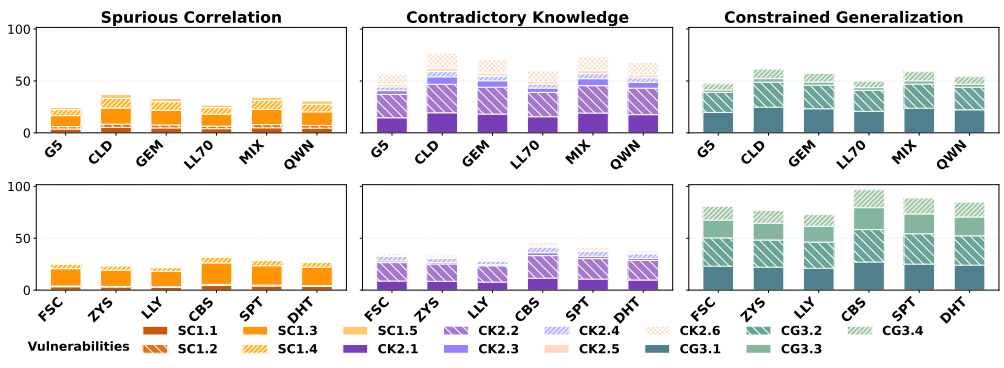


Figure 3: Varying proportions of failure modes (types listed in Figure 1). Note that different failure modes can intertwine within the same instance, which is particularly common in Contradictory Knowledge (CK) and Constrained Generalization (CG), less common in Spurious Correlation (SC).

volume of failures. This indicates that their broad but non-specialized training leaves them vulnerable to overfitting and misinterpreting contextual evidence.

In contrast, for failure modes rooted in the fine-tuning data (e.g., confounding factors in the threat corpus), cyber agents such as Foundation-Sec tend to produce less reliable outputs (e.g., forecasting exploitability with contradictory PoCs). Such failure modes act as data poisoning, where spurious correlations and contradictory knowledge can arise either intentionally (introduced by adversaries) or unintentionally (inherent in the fragmented cyber threat landscape).

Besides, we also observed that all models show constrained performance when confronted with emerging or zero-day threats. Failure ratios are particularly high in specialized cyber agents, whose localized nature and narrower pre-trained knowledge bases limit their adaptability. Once their training cutoffs are reached, they lack the generalization capacity to extrapolate effectively to novel threats.

G.3 Root Cause Analysis

This part provides additional and exhaustive root cause analysis across each failure mode type in different CTI stages, which complements §4.2.

Co-mention bias (Type 1.1) in ❶ contextualization. This part has been introduced in §4.2.

Co-mention bias (1.1) in ❷ attribution. In attribution, co-mention bias presents when reports describe overlapping infrastructure or techniques across multiple actors. A single campaign report may reference domains, malware families, or TTPs associated with different groups, not because of true collaboration but due to co-reporting practices. Relation extraction or graph construction techniques then misinterpret these shared mentions as evidence of actor overlap or shared lineage, resulting in erroneous attribution. For instance, if infrastructure reused by both APT X and APT Y is co-mentioned, the model may incorrectly assign responsibility to one actor or merge distinct campaigns. The root cause is that attribution pipelines often lack mechanisms to filter incidental co-mentions from true operational reuse.

Case Study: Co-mention bias (1.1) in ❷ attribution.

As reported by Group-IB (Tok & Celiktas, 2019), analysts observed infrastructure overlap in domains or servers co-mentioned across multiple campaigns attributed to MuddyWater. However, further investigation revealed that although the same infrastructure was used or appeared in reports, the operational characteristics (targeting, malware payloads, attack vectors) differed significantly between those campaigns. Because relation extraction or graph building techniques often rely on co-occurrence of infrastructure to infer actor reuse, models might erroneously link distinct campaigns to the same threat actor, or assume a shared campaign lineage, solely based on the shared infrastructure.

Co-mention bias (1.1) in ❸ prediction. In prediction tasks, co-mention bias arises when models forecast exploit likelihood or campaign escalation based on correlated but unrelated evidence in prior reports. For example, if a dataset frequently co-mentions a vulnerability with a high-profile

exploit alongside unrelated low-severity flaws, predictive models may overestimate the risk of the latter. Similarly, temporal modeling that uses co-occurring events may incorrectly infer progression paths between independent threat activities. Historical event correlation amplifies this effect: threats repeatedly co-mentioned in vendor advisories may be predicted to evolve together, even when no causal relationship exists. Here, the root cause is an overreliance on surface-level temporal or co-occurrence signals without causal disentanglement.

Co-mention bias (1.1) in ❹ mitigation. In mitigation, co-mention bias leads to flawed defensive recommendations. CTI reports often list multiple countermeasures (e.g., patches, YARA rules, firewall configurations) together, though not all apply to a given threat instance. Mitigation mapping systems that rely on keyword overlaps may thus associate irrelevant countermeasures with observed TTPs, producing noisy or infeasible recommendations. For example, if two patches are co-mentioned in an advisory but only one addresses the exploited vulnerability, the model may still rank both as equally necessary. Similarly, summarization pipelines can inadvertently include unrelated mitigations, generating bloated or misaligned playbooks. The root cause lies in the assumption that co-mentioned countermeasures share equal applicability, ignoring the fine-grained specificity required in operational defenses.

Case Study: Co-mention bias (1.1) in ❹ mitigation.

In Microsoft's June 2025 Patch Tuesday advisory, 65 CVEs were addressed, including a zero-day vulnerability exploited in the wild. The advisory lists patches for many components including Microsoft Office, Visual Studio, Windows Kernel, WebDAV, SMB, and others. Because patching is broad and multiple fixes are mentioned in the same bulletin, a CTI model might treat all listed patches as equally urgent defense actions — even though only a small subset correspond to vulnerabilities currently exploited. This leads organizations or systems to over-allocate resources toward non-critical patching, or generate mitigation playbooks that include countermeasures not immediately relevant to active threats.

Exploitation bias (1.2) in ❶ contextualization. In contextualization, exploitation bias emerges when reused IoCs (e.g., IPs, domains, hashes) are repeatedly referenced across unrelated incidents, leading enrichment pipelines to overgeneralize their significance. Information retrieval systems may surface threat reports where the same IP is mentioned in multiple campaigns, without clarifying whether it is genuinely reused by adversaries or simply a shared infrastructure artifact (e.g., cloud hosting or CDN services). Topic modeling and knowledge base mapping exacerbate this issue by clustering these co-occurrences, causing LLMs to link disparate events as if they were causally connected. The root cause here is that raw IoC reuse lacks contextual disambiguation, so enrichment steps propagate spurious relevance signals that distort downstream reasoning.

Exploitation bias (1.2) in ❷ attribution. Threat actor identification often relies on detecting overlaps in IoCs, assuming that infrastructure reuse reflects shared adversary control. However, many IoCs are deliberately reused by attackers to create ambiguity, or coincidentally shared due to compromised hosting providers. Relation extraction and event graph construction may then over-attribute distinct campaigns to a single actor, collapsing multiple adversarial lineages into one. For instance, if the same command-and-control domain appears across incidents attributed separately to APT28 and APT29, attribution models may incorrectly merge them. The root cause is an assumption of exclusivity in IoC ownership, which adversaries exploit through deliberate recycling of infrastructure.

Exploitation bias (1.2) in ❸ prediction. During prediction, reused IoCs bias forecasts by inflating the perceived likelihood of exploitation or campaign escalation. Historical event correlation and temporal modeling often treat repeated appearances of an IoC as evidence of persistent activity, projecting elevated future risk. Yet, in reality, the reuse may reflect low-cost attacker behavior (spamming multiple targets with the same domain) rather than meaningful escalation. Graph-based forecasting amplifies this, propagating edges from over-represented IoCs across multiple vulnerability nodes, leading to inflated EPSS-like scores. The root cause lies in predictive models' reliance on frequency of appearance, without mechanisms to discount intentionally or incidentally reused IoCs.

Case Study: Exploitation bias (1.2) in ❸ prediction.

Multiple CVEs share many IoCs (e.g., IP addresses and domain names) across different CTI provider feeds (Kodituwakku et al., 2023). Over time, some of these IoCs appear repeatedly in contexts of various vulnerabilities, even though not all of them are actually exploited in relation to each CVE. Because prediction models often take frequency of IoC appearance as a strong signal, they tend to assign higher risk to these

CVEs or predict escalation based on the reused IoCs. In this way, reuse of IPs/domains (which may simply reflect broad surveillance coverage or shared infrastructure, not genuine exploit activity) inflates perceived future threat likelihoods.

Exploitation bias (1.2) in ❹ mitigation. In mitigation, exploitation bias presents when countermeasure recommendations are tied too strongly to reused IoCs. For example, signature generation systems may repeatedly create YARA rules around the same recycled domain or hash, producing redundant or noisy detection logic. Patch recommendation pipelines may mis-prioritize vulnerabilities linked to reused IoCs, assuming their recurrence reflects higher severity. Summarization systems generating response playbooks may include repeated references to blocking the same IoC across different incidents, inflating defensive burden without increasing actual protection. The root cause is that mitigation mapping often equates frequency of IoC appearance with actionable importance, overlooking the adversary tactic of deliberate recycling.

Confounding factors (1.3) in ❶ contextualization. In contextualization, confounding factors often emerge when raw observations contain multiple entities that correlate implicitly but lack a direct causal relationship. For instance, topic modeling applied to large incident corpora may cluster a vulnerability (CVE) with a malware family simply because they are frequently mentioned together in reports, even if the malware never exploited that vulnerability. Similarly, event extraction can incorrectly bind unrelated infrastructure (e.g., a benign domain) to a malicious timeline because both appear in the same paragraph. The root cause is that contextual enrichment techniques rely heavily on co-occurrence or textual proximity, which implicitly correlates entities without accounting for deeper causal validation, thereby inflating the contextual landscape with misleading links.

Confounding factors (1.3) in ❷ attribution. Confounding factors are especially problematic in attribution, where analysts and models attempt to connect behaviors to actors. Relation extraction and event graph construction can erroneously merge distinct campaigns if they share surface features — for example, multiple threat groups may use commodity malware or overlapping hosting providers. Without careful disentanglement, the attribution pipeline interprets these shared attributes as strong evidence of common authorship. Moreover, stylistic signals such as language or compilation timestamps may correlate with regional actors but can be misleading when adversaries deliberately obfuscate or mimic others. Thus, attribution systems are vulnerable to implicit correlations that misguide actor classification, producing overconfident but flawed linkages between incidents and threat groups.

Case Study: Confounding factors (1.3) in ❷ attribution.

In our attribution dataset, we observed a campaign exploiting **CVE-2022-1388** (a remote code execution vulnerability in F5 BIG-IP devices) that was mistakenly clustered with another intrusion attributed to a different actor. Both campaigns used the same publicly available exploitation script and temporarily shared IP infrastructure via a bulletproof hosting provider. Although the payloads and target industries differed, the attribution pipeline—heavily relying on shared malware hashes and infrastructure proximity—merged the two incidents under a single actor label. Further analysis revealed that one group had intentionally mimicked the operational cadence and header patterns of the other, introducing stylistic confusion. This misattribution was rooted in the model's inability to disentangle commodity tooling from actor-specific behavior, exemplifying how confounding factors can mislead attribution systems.

Confounding factors (1.3) in ❸ prediction. During prediction, confounding factors distort forecasting by elevating signals that are correlated with exploitation or escalation but not truly causal. For example, historical event correlation may reveal that vulnerabilities discussed in the same advisories as high-severity flaws appear more likely to be exploited, even if they are rarely targeted in practice. Time series models may overweight recurring co-mentions across campaigns, predicting that certain malware–sector combinations will reappear simply because of their past textual co-occurrence. Graph neural networks, when trained on CTI event graphs, may propagate spurious links (e.g., connecting two vulnerabilities through a shared but irrelevant indicator), reinforcing false associations. Here, the root cause lies in the inability of predictive models to filter out spurious correlates from genuine causal drivers.

Confounding factors (1.3) in ❹ mitigation. In mitigation, confounding factors lead to noisy or misaligned defensive recommendations. For instance, mitigation mapping systems may associate a patch with multiple unrelated TTPs simply because they were mentioned in the same advisory, conflating the true scope of the fix. Similarly, defensive playbook generation may cluster unrelated

countermeasures together, producing bloated response strategies that overprescribe actions. Even mitigation efficacy prediction models can be misled if training data shows that certain mitigations are frequently co-listed with high-profile vulnerabilities, causing the model to rank them as universally effective. The underlying issue is that mitigation pipelines frequently assume that correlated mentions of threats and countermeasures imply operational relevance, leading to inflated or misdirected defense guidance.

Skewed source (1.4) in ❶ contextualization. Skewed source bias originates primarily in contextualization because this stage depends heavily on external retrieval and enrichment pipelines. When information retrieval systems disproportionately pull from certain vendors, open-source repositories, or community feeds, the resulting knowledge base becomes skewed toward specific regions, vendors, or product lines. For instance, a RAG system trained on CTI feeds dominated by North American vendors will overrepresent CVEs affecting widely deployed enterprise systems, while underrepresenting region-specific threats or mobile malware in other ecosystems. Topic modeling and event extraction then propagate this imbalance by clustering narratives around the most frequently indexed vendors rather than providing a representative view of the global threat landscape. Thus, the root cause lies in uneven data source availability and ingestion pipelines, which distort the contextual foundation upon which downstream reasoning is built.

Case Study: Skewed source (1.4) in ❶ contextualization.

In our contextualization experiments, we analyzed enrichment results for **CVE-2023-20963** (a critical privilege escalation vulnerability affecting Google Pixel devices). While regional CERTs and Android security blogs had documented active exploitation of this CVE in Southeast Asia, our retrieval module—trained predominantly on English-language feeds from North American enterprise vendors—failed to surface these reports. As a result, the contextualization pipeline linked the CVE only to generic kernel privilege escalation patterns, without associating it with active campaigns or mobile-targeted payloads. Topic modeling then grouped the CVE under server-side vulnerabilities rather than mobile device threats, further distancing it from relevant mitigation data. This illustrates how over-reliance on skewed sources can suppress visibility into region-specific threats and impair downstream enrichment quality.

Skewed source (1.4) in ❷ attribution (Why not influential). Attribution is less directly affected by skewed source bias, since once contextualized entities are available, the task focuses on linking them to adversary profiles or campaigns. While attribution accuracy can degrade if upstream contextualization is biased, the attribution process itself (e.g., relation extraction, event graph construction, stylistic profiling) does not inherently depend on the relative volume of one vendor's reports over another. Instead, attribution errors are more likely to stem from confounding overlaps, contradictory knowledge, or overfitted reasoning. Therefore, skewed source bias has only an indirect influence at this stage, primarily through the quality of contextual signals passed forward.

Skewed source (1.4) in ❸ prediction (Why not influential). Prediction tasks such as exploit likelihood estimation, campaign escalation modeling, or sectoral targeting forecasts typically rely on historical event correlation and temporal modeling, rather than raw source diversity. Once contextualized and attributed data is available, predictive models infer temporal or causal structures independent of whether one vendor dominates the input streams. Skewed sources may still indirectly shape the training distribution (e.g., overpredicting risks for vendor-popular products), but the predictive mechanisms themselves are not fundamentally triggered by source skew. Instead, failures in this stage more commonly arise from unseen patterns, zero-day threats, or distributional bias in event histories rather than skewed input sources.

Skewed source (1.4) in ❹ mitigation (Why not influential). Mitigation is similarly insulated from direct skewed source effects. Once recommendations are mapped (e.g., patches, rules, or playbooks), the ranking and summarization steps focus on aligning countermeasures with observed TTPs or vulnerabilities, not on the origin of the source reports. While an upstream bias may have limited the initial diversity of vulnerabilities considered, the mitigation stage itself does not amplify source skew. Errors in mitigation typically reflect misaligned standards (e.g., CVSS vs vendor scoring), counteracted evidence (patch bypasses), or environmental unawareness (system-specific configurations). Thus, skewed source remains a contextualization-stage vulnerability whose downstream effects are secondary rather than intrinsic to mitigation logic.

Hierarchical metadata (1.5) in ❶ contextualization. During contextualization, hierarchical metadata embedded in attack chains can mislead enrichment processes. MITRE ATT&CK or similar

frameworks present TTPs as sequences in which adversaries may progress from initial access to impact. However, when LLMs or RAG-based pipelines ingest this structured knowledge, they may mistakenly treat the ordering as deterministic rather than illustrative. For example, if multiple unrelated IOCs are linked to a chain stage, the model may infer that they are causally connected because of their shared position in the hierarchy. Topic modeling and event extraction further amplify this issue, since co-occurrence within the same hierarchical step often gets interpreted as functional equivalence, thereby creating spurious associations between distinct threat activities. The root cause is the conflation of descriptive, taxonomy-based ordering with ground-truth causal relations.

Hierarchical metadata (1.5) in ❷ attribution. In attribution, hierarchical metadata bias presents when structured attack chain taxonomies are used to connect adversary behavior to actor profiles. Threat reports often highlight sequences of TTPs that adversaries are "known" to employ, but in practice attackers skip, reorder, or substitute steps. Relation extraction and graph construction tools, however, may rigidly map observed behavior to the canonical hierarchy, leading to over-attribution. For instance, if two groups share overlapping steps in the ATT&CK chain (e.g., persistence or lateral movement), hierarchical metadata can cause models to collapse them into the same attribution cluster, even if their infrastructure and operational cadence differ. Thus, reliance on hierarchical metadata in attribution conflates broad behavioral categories with actor-specific evidence.

Case Study: Hierarchical metadata (1.5) in ❷ attribution.

In our attribution experiments, we observed a misclassification related to **CVE-2022-30190** (Follina Microsoft Support Diagnostic Tool RCE), where two distinct campaigns—one leveraging Microsoft Office documents, and another using HTML smuggling techniques—were both partially mapped to the same sequence in the MITRE ATT&CK framework (Initial Access → Execution → Lateral Movement). Despite clear differences in command-and-control infrastructure and target sectors, the graph construction module merged both campaigns under a single actor cluster due to their alignment with a common TTP hierarchy. The system interpreted the overlapping ATT&CK phases as evidence of a shared operational lineage. However, manual review confirmed the two campaigns were launched by different groups, with different goals and temporal scopes. This case illustrates how over-reliance on hierarchical metadata can collapse distinct operations into the same attribution cluster, reducing fidelity.

Hierarchical metadata (1.5) in ❸ prediction. Prediction tasks are particularly vulnerable to hierarchical metadata bias, as temporal models often use sequential patterns from attack chains to forecast campaign escalation. Forecasting tools may assume that once a threat is observed at one stage (e.g., privilege escalation), subsequent hierarchical steps (e.g., data exfiltration) will necessarily follow. This prescriptive interpretation leads to inflated probabilities of certain outcomes, even when the adversary's campaign objectives differ. For example, opportunistic attackers may terminate activity after initial access without advancing through the full chain, but predictive models, trained on hierarchical metadata, extrapolate full kill chain completion. Here the root cause is the overgeneralization of taxonomy-driven sequences as predictive signals of adversarial intent.

Hierarchical metadata (1.5) in ❹ mitigation (why not influential). By contrast, mitigation tasks are less directly affected by hierarchical metadata bias. Defensive actions such as patch application, YARA rule generation, or firewall tuning are typically grounded in concrete IOCs or known vulnerabilities rather than inferred positions in an attack chain. Mitigation mapping focuses on linking observed TTPs to available defensive strategies, not on reconstructing or extrapolating hierarchical steps. Even if upstream contextualization or prediction stages have been biased, mitigation operates on a more pragmatic level: "given X IOC or Y vulnerability, recommend Z patch or rule." Thus, hierarchical metadata plays a minimal role in this stage, since response generation depends on actionable artifacts rather than assumed causal ordering of adversarial behaviors.

Temporal contradiction (2.1) in ❶ contextualization. In contextualization, temporal contradictions emerge when retrieval or enrichment systems ingest both outdated and recent advisories without properly disambiguating their validity. For instance, information retrieval pipelines may surface a vendor's original advisory that labeled a vulnerability as "under investigation" alongside a newer update stating it has been patched. Topic modeling or knowledge base mapping then treat both pieces of information as equally valid, causing LLMs to enrich raw indicators with conflicting metadata. This leads to enriched contexts where a vulnerability is simultaneously "exploited in the wild" and "not yet confirmed," which can misguide subsequent correlation. The root cause is the absence of temporal weighting or source freshness filters in contextualization pipelines.

Temporal contradiction (2.1) in ❷ attribution. In attribution, temporal contradictions present when different reports about an adversary's activity span different time periods but are fused as if they describe the same campaign. Relation extraction or event graph construction may link infrastructure referenced in an outdated report to more recent attack chains, even if the adversary has long since abandoned those assets. Similarly, named entity recognition can tag an actor profile based on old malware usage, which conflicts with newer intelligence indicating a complete toolset shift. This results in inflated or inaccurate attribution, where models mistakenly conclude that an actor is reusing infrastructure or TTPs when the overlap exists only across time-separated reports. The root cause here is insufficient temporal resolution in attribution graphs and classification models.

Temporal contradiction (2.1) in ❸ prediction. Temporal contradictions pose particular risks in prediction, where forecasts rely heavily on historical data. If older datasets mark a vulnerability as non-exploitable, but newer advisories confirm widespread weaponization, models trained on both may generate unstable exploitability estimates. Temporal modeling pipelines may inadvertently treat early false negatives and later confirmed positives as equivalent, confusing progression trends. Similarly, forecasting models using historical event correlation may blend outdated infrastructure associations with current actor behaviors, leading to incorrect campaign evolution predictions. The root cause lies in the lack of mechanisms to discount superseded evidence, which creates contradictory temporal signals during exploit likelihood and impact modeling.

Case Study: Temporal contradiction (2.1) in ❸ prediction.

A concrete example is the evolution of exploit activity for **CVE-2020-1472 (Zerologon).** Initially, after its disclosure, exploit usage was limited and many datasets treated the vulnerability as low risk. Over time, threat actors began leveraging Zerologon widely, including in ransomware and lateral movement toolchains. EclecticIQ's long-term analysis demonstrates how risk associated with this vulnerability evolved substantially across months, with early scarce activity later giving way to broad exploit adoption. Predictive models that do not de-emphasize earlier false negatives risk underestimating exploitability or generating contradictory forecasts. This case underscores how temporal contradiction — mixing outdated low-risk labeling with current high exploitation data — can destabilize predictions of vulnerability exploit likelihood.

Temporal contradiction (2.1) in ❹ mitigation. During mitigation, temporal contradictions can mislead response prioritization. For example, mitigation mapping may retrieve a vendor patch note stating "fix not yet available," while a newer advisory lists a released patch. Summarization or mitigation ranking systems that fail to resolve the temporal sequence may present both claims to the analyst, leaving uncertainty about whether a defense is actionable. Similarly, mitigation efficacy prediction models may weigh outdated proofs-of-concept that show patch bypasses against more recent vendor confirmations that the patch has been hardened, producing contradictory rankings. The root cause is inadequate reconciliation of evolving patch and mitigation information over time, which allows old and obsolete advice to persist alongside updated guidance.

Conflicting report (2.2) in ❶ contextualization. Conflicting reports frequently arise during contextualization because raw observations are aggregated from diverse sources (e.g., vendor advisories, community threat feeds, and open reports) that describe incidents with varying detail and emphasis. For example, one feed may assert that a suspicious log entry corresponds to exploitation of a particular CVE, while another links the same log to an entirely different vulnerability or malware family. Topic modeling and event extraction pipelines, designed to cluster narratives and structure incidents, may then merge these contradictory claims into a single enriched context. This creates confusion over which CVE, TTP, or malware family is actually relevant, with the root cause being the absence of a standardized ground truth and the reliance on partially overlapping but inconsistent external reports.

Case Study: Conflicting report (2.2) in ❶ contextualization.

In our contextualization dataset, we observed a case related to **CVE-2020-1472** (Netlogon Elevation of Privilege) where two intelligence sources described the same suspicious authentication anomaly differently. Source A associated the log entries with attempted exploitation of CVE-2020-1472 based on domain controller access patterns, while Source B linked the same entries to a credential stuffing attack referencing a password reuse vulnerability. Our event extraction and clustering module merged both narratives into one incident node, assigning mixed attributes (some from CVE-2020-1472 context, some from credential reuse context). As a result, downstream enrichment conflated the CVE and misattributed the observed behavior to the wrong TTP/malware family. This demonstrates how conflicting reports can introduce ambiguity in contextualization, leading to corrupted or merged contexts that mislead later modules.

Conflicting report (2.2) in ❷ attribution. In attribution, contradictions become even more pronounced because different intelligence providers often assign conflicting actors, dependencies, or motivations to the same activity. For instance, one report may classify an intrusion as the work of APT29 based on shared TTPs, while another attributes the same infrastructure to APT28 due to linguistic or operational cadence evidence. Relation extraction and graph construction techniques that link entities across sources may therefore inherit and fuse these contradictions, producing noisy actor graphs or overextended campaign links. The underlying root cause is the subjectivity and methodological diversity of attribution across organizations — some weigh infrastructure overlaps more heavily, others focus on malware lineage — which produces irreconcilable dependency paths when combined.

Conflicting report (2.2) in ❸ prediction. Prediction models also suffer from contradictory reports when forecasting exploitation likelihood or campaign escalation. Historical event correlation may treat conflicting dependency chains as equally plausible futures: for instance, one source suggests a vulnerability will be exploited in ransomware campaigns, while another asserts it is tied to espionage-focused actors. Temporal modeling then captures both trajectories, leading to unstable or diluted predictions. Graph neural networks used for campaign evolution may amplify these conflicts, generating escalation paths that reflect contradictory dependencies across threat actors or malware ecosystems. The root cause lies in how predictive systems implicitly assume coherence in historical data, but when inputs encode conflicting dependencies, the forecasts inherently inherit those contradictions.

Conflicting reports (Type 2.2) in ❹ mitigation. Mitigation guidance often diverges across vendors due to differences in testing environments, threat visibility, and evaluation scope. For example, one advisory may claim a patch fully addresses an exploited vulnerability, while another provides evidence of continued exploitation via chained dependencies (e.g., an adjacent misconfiguration). Mitigation mapping built on such contradictory sources can overestimate or underestimate the effectiveness of a countermeasure. Similarly, summarization pipelines may produce inconsistent playbooks (e.g., some prioritizing patch deployment and others emphasizing compensating controls) leaving LLMs uncertain about actions truly mitigate the incident.

Case Study: Conflicting reports (Type 2.2) in ❹ mitigation.

In the 2021 *Microsoft Exchange "Hafnium"* case (Pitney et al., 2022), some advisories claimed patches fully mitigated the threat, while others warned of persistent web shells post-patch. The conflicting reports gave Qwen retriever false assurance about their mitigation effectiveness.

Semantic conflict (2.3) in ❶ contextualization. Semantic conflict most prominently arises during contextualization because this stage depends heavily on mapping raw threat indicators to standardized taxonomies such as CVEs, MITRE ATT&CK TTPs, or malware family names. In practice, different vendors or intelligence feeds often use divergent terminology for the same underlying concept — for example, one feed may classify an intrusion set as "APT28," while another uses "Fancy Bear." Similarly, malware families may appear under multiple aliases (e.g., "PlugX" versus "Korplug"), or distinct CVEs may be referenced with vendor-specific identifiers. When contextualization systems use topic modeling, event extraction, or knowledge base alignment, they may fail to reconcile these terminological mismatches. This results in fragmented knowledge graphs where semantically identical entities are treated as distinct, leading to duplicated or incomplete enrichment. The root cause is the reliance on surface-level string matching or incomplete ontology alignment when normalizing threat information across heterogeneous data sources.

Case Study: Semantic conflict (2.3) in ❶ contextualization.

In our contextualization pipeline logs, we observed a case related to **CVE-2022-22965** (Spring4Shell remote code execution vulnerability) in which two CTI sources referred to the underlying exploit using different aliases: one feed tagged the exploit as "SpringShell," while another labeled it "Spring4Shell RCE." Because our mapping module used strict string-based matching and a limited alias dictionary, it failed to align the two aliases to the same canonical exploit entity. As a result, enrichment modules treated the two references separately, creating two parallel nodes in the knowledge graph with only partial contextual links (one with attacker metadata, the other with domain reuse). Downstream modules thus had incomplete context when correlating indicators across both nodes, weakening detection or attribution inference. This illustrates how semantic conflict in naming can fragment context and destabilize CTI contextualization outputs.

Semantic conflict (Type 2.3) in ❷ attribution (less influential). Semantic conflict is less influential in attribution because actor identification depends on relational evidence and behavioral signatures. In practice, attribution relies more on infrastructure reuse, campaign TTPs, and stylistic cues than on whether a malware family is labeled "PlugX" or "Korplug." Relation extraction and event-graph construction emphasize these structural patterns, making attribution models relatively tolerant to naming inconsistencies as long as the underlying relationships remain stable. While semantic divergence can introduce minor noise, it rarely flips the actor assignment: different labels for the same family typically still connect to the same infrastructure nodes and TTPs. As a result, attribution is comparatively robust to semantic conflicts.

Case Study: Semantic conflict (Type 2.3) in ❷ attribution (less influential).

Early CTI reporting showed **contextualization failure (Type 1.3)**: triage lumped the SQL injection into generic IIS exploitation, and *co-mention bias* hid the specific bug in `moveitisapi.dll`. Later, **Attribution failure (Type 2.1)** misattributed the activity to *Silence* due to the "Truebot" downloader, despite later evidence pointing solely to *Cl0p (Lace Tempest)*. Finally, **Prediction failure (Type 3.2)** underestimated the *Impact Forecast*, missing the supply-chain cascade (e.g., Zellis, BBC) because the model overgeneralized from typical single-victim ransomware cases.

Semantic conflict (2.3) in ❸ prediction (Why not influential). Prediction tasks are not substantially affected by semantic conflict because forecasting relies on temporal dynamics and statistical trends rather than entity naming conventions. For example, exploit likelihood estimation (e.g., EPSS) depends on historical exploitation rates, vulnerability characteristics, and observed attack timelines. Whether a malware strain is labeled with one alias or another has little bearing on probability estimates, since the predictive models aggregate numerical and structural features rather than semantic labels. Similarly, temporal modeling and graph neural networks capture correlations between event sequences independent of specific vocabulary. As a result, semantic inconsistencies have negligible impact on predictive accuracy, except in rare cases where mislabeled data significantly distorts training distributions.

Semantic conflict (2.3) in ❹ mitigation (Why not influential). Mitigation is minimally impacted by semantic conflict, because defensive actions map to vulnerabilities, TTPs, or attack surfaces rather than to naming conventions alone. For example, a patch recommendation system aligns a CVE identifier with its associated vendor fix, regardless of whether different advisories describe the vulnerability with varied terms. Similarly, YARA rule generation focuses on technical indicators such as byte patterns or log events, which are invariant to naming disputes. Response playbooks are typically tied to standardized defensive frameworks (e.g., ATT&CK, CVSS), which already normalize naming variations. While semantic divergence may cause minor confusion in documentation or cross-team communication, it rarely degrades the technical quality of mitigation outputs.

Divergent structures (2.4) in ❶ contextualization. In contextualization, divergent data structures create noise when raw threat intelligence is aggregated from multiple heterogeneous sources such as JSON-based threat feeds, PDF advisories, STIX-formatted indicators, and unstructured blog posts. Information retrieval and knowledge base mapping pipelines often assume consistent schema alignment, yet structural discrepancies (e.g., different field names for "affected system," varying timestamp formats, or nested vs. flat representations of IoCs) cause mismatches. When LLMs enrich observations under these inconsistencies, some attributes may be duplicated, dropped, or misinterpreted. For instance, one platform may list malware family as a top-level attribute while another embeds it in narrative text, leading to incomplete enrichment. The root cause is the lack of robust schema normalization during the fusion of structurally diverse CTI inputs.

Divergent structures (2.4) in ❷ attribution. Attribution pipelines rely on entity and relation extraction, but divergent data structures across platforms distort graph construction. Structured feeds may represent relationships (e.g., IP–Domain–Actor) as explicit triples, while unstructured reports only provide natural language references. Relation extraction systems trained on one schema may fail on the other, creating inconsistent or fragmented event graphs. This can yield incorrect attributions, such as splitting a single campaign into multiple unrelated clusters or merging distinct actor profiles due to structurally divergent naming conventions. For example, some feeds may encode actor aliases explicitly, while others bury them in footnotes, leading the model to under- or over-aggregate. The root cause is schema heterogeneity across platforms that undermines consistent relation representation.

Case Study: Divergent structures (2.4) in ❷ attribution.

In our attribution dataset, we observed a case involving **CVE-2019-0708** (BlueKeep Remote Desktop Services vulnerability) where structural heterogeneity between a structured threat feed and a narrative security report caused contradictory graph links. Specifically, a structured CTI feed represented the relation "Actor Delta uses DomainX → IPY" as a neat triple, allowing direct linking to that actor. Meanwhile, a free-text incident report described the same infrastructure but only said "the attacker used a domain resolved via IPY, associated with DomainX, commonly tied to Delta's campaigns" in a footnote. The relation extraction model, trained primarily on structured triple formats, failed to recognize the footnote phrasing as linking to Actor Delta, instead treating it as an independent mention. As a result, the domain/IP chain was disconnected in the event graph, causing the system to assign the activity to a generic "Unattributed" cluster rather than merging with Delta's campaign. In another variant, when domain aliasing was encoded differently, the system mistakenly merged it with Actor Epsilon, whose structured feed used a similar alias triple, thereby conflating two distinct actors. This demonstrates how divergent structural formats across platforms can fragment or overmerge attribution graphs in real-world CTI pipelines.

Divergent structures (2.4) in ❸ prediction. Prediction systems that depend on temporal modeling or cross-platform event correlation are particularly sensitive to structural divergence. Historical correlation models assume standardized event logs, but if one feed provides vulnerability exploitation dates as free-text while another encodes them as epoch timestamps, time-series models may misalign or discard the data. Similarly, graph neural networks forecasting campaign escalation require uniform edge types and attributes; divergent structures across CTI sources (e.g., "sector" encoded as a categorical variable in one feed but as descriptive text in another) disrupt model training and inference. This leads to biased forecasts or missed escalation patterns. The root cause is that predictive algorithms cannot reconcile structurally inconsistent features, weakening their ability to capture true temporal and relational dynamics.

Divergent structures (2.4) in ❹ mitigation (Why not influential). Mitigation tasks, unlike the earlier stages, are relatively insulated from divergent data structure issues because defensive actions tend to be represented in standardized, prescriptive formats. Patches are tied to CVE identifiers, detection rules often use formal languages like YARA, and configuration changes are usually documented as explicit command lines or policy instructions. These standardized forms limit the degree of structural variation compared to raw threat intelligence data. While minor discrepancies may occur (e.g., different vendors labeling patch IDs differently), the core content is highly structured and task-specific, reducing the likelihood that divergent data structures across platforms significantly affect mitigation outputs.

Misaligned standards (2.5) in ❶ contextualization (Why not influential). In the contextualization stage, misaligned knowledge and security standards generally do not exert a strong influence. The task here is primarily to enrich raw observations (logs, IOCs, alerts) with contextual information such as CVEs, malware families, and ATT&CK TTPs. While standardization issues exist in naming conventions, they do not typically fall under formal security scoring or prioritization frameworks. Thus, contextualization is more affected by spurious correlation or semantic conflicts rather than by misaligned standards, since the goal is mapping and enrichment rather than prioritization or scoring.

Misaligned standards (2.5) in ❷ attribution. Misaligned standards can distort attribution outcomes when different organizations adopt varying taxonomies or classification schemes for actors and campaigns. For example, one CTI provider may label a campaign under an APT designation (e.g., "APT29"), while another refers to the same activity under a vendor-specific alias (e.g., "Cozy Bear"). When models integrate these heterogeneous standards, they may fragment evidence across labels or mistakenly merge distinct actors. Similarly, divergence in the way behavioral profiles or TTPs are scored—some focusing on tool use, others on infrastructure overlap—can misguide attribution classification. The root cause lies in the absence of universally accepted standards for actor naming and behavior profiling, which introduces structural misalignment into attribution models.

Misaligned standards (2.5) in ❸ prediction. In prediction, misaligned knowledge and standards significantly amplify failure. Forecasting tasks often rely on vulnerability severity ratings (e.g., CVSS), exploit prediction scores (e.g., EPSS), or proprietary vendor risk models. A misalignment occurs when different standards assign conflicting severities to the same vulnerability: one database might rank it as "Critical" due to remote code execution potential, while another rates it "Medium" because of authentication requirements. Temporal modeling or forecasting systems ingesting both signals may oscillate between divergent risk profiles, leading to unstable exploitability predictions or

over/under-estimation of campaign escalation risks. The core issue is that prediction models assume commensurability of scores, when in reality, standards reflect different prioritization philosophies.

Case Study: Misaligned standards (2.5) in ❸ prediction.

In our prediction logs, we observed an instance involving **CVE-2021-26855** (Exchange Server ProxyLogon vulnerability) where different scoring sources conflicted sharply. One threat feed assigned it a Critical severity in CVSS (base score 9.x) given its remote code execution nature, while another vendor's internal risk model (factoring in required authentication, exploit maturity, and environment heuristics) gave it only a Medium rating. When our prediction module ingested both signals, it produced unstable forecasts: in some runs it predicted high likelihood of exploitation, and in others it down-prioritized the CVE, delaying alert escalation. Further review showed active exploitation in the wild shortly thereafter, confirming the "Critical" perspective. This case underscores how misaligned standards induce unstable predictions by conflicting risk signals in CTI-driven forecasting.

Misaligned standards (2.5) in ❹ mitigation. Mitigation is perhaps the most directly affected by misaligned standards. Defensive recommendations often depend on mappings between observed TTPs, vulnerabilities, and standardized mitigation catalogs (e.g., NIST, MITRE ATT&CK mitigations, vendor advisories). When standards differ (such as one framework emphasizing patching order by CVSS scores, while another emphasizes sector-specific asset criticality) conflicting guidance arises. For instance, a vulnerability may be labeled as high-priority patching in NVD, but a vendor advisory may downplay its urgency, creating contradictory recommendations for SOC teams. Summarization or playbook generation techniques then risk producing inconsistent or misleading instructions, either overwhelming defenders with unnecessary actions or under-preparing them for critical exploits. The root cause is the lack of harmonization between security scoring systems and defensive taxonomies, which propagates inconsistencies directly into operational decision-making.

Counteracting generation (2.6) in ❶ contextualization (Why not influential). Counteracting CTI generation and LLM alignment has minimal influence at the contextualization stage. Here, the primary task is enrichment — mapping raw observations (e.g., logs, IOCs, alerts) to known identifiers like CVEs or ATT&CK TTPs. Because this process relies largely on retrieval, topic modeling, and knowledge base mapping, contradictions across sources tend to surface as co-mention bias or temporal conflict rather than as direct misalignment of the model's reasoning process. LLMs can still ground outputs in retrieved evidence, even if that evidence is noisy. In other words, contextualization errors usually inflate or distort context but rarely cause the model's generation logic itself to become unstable or self-counteracting.

Counteracting generation (2.6) in ❷ attribution. In attribution, counteracting CTI generation manifests strongly. Contradictory or conflicting reports of actor identities, infrastructure reuse, or campaign affiliations directly challenge the alignment of an LLM fine-tuned to classify or link entities. For instance, one source may attribute an intrusion set to APT29, while another insists on APT28, forcing the LLM to reconcile irreconcilable evidence. During training, these conflicting signals weaken gradient alignment, creating internal tension where the model oscillates between incompatible actor labels. At inference time, relation extraction or event graph construction may generate unstable outputs — e.g., merging distinct campaigns into a single actor cluster, or switching attribution mid-response. The root cause lies in the inability of LLM alignment processes to disentangle contradictory ground truths when adversarial or incomplete evidence coexists in CTI corpora.

Case Study: Counteracting generation (2.6) in ❷ attribution.

In our attribution dataset, we observed a conflict surrounding **CVE-2020-0796** (SMBGhost / SMBv3 remote code execution vulnerability) where two report clusters described overlapping infrastructure and payloads but assigned divergent actor labels. One internal CTI source traced the campaign to **Actor Beta** based on reused command-and-control domain naming conventions, while another equally plausible source assigned it to **Actor Gamma** citing similar malware module signatures. When an LLM-based attribution module attempted to reconcile the evidence, it oscillated between Actor Beta and Actor Gamma at different points in the generated response, and in one case merged both actor clusters into a single ambiguous actor node. Forensic cross-checks revealed that the two clusters employed distinct lateral propagation chains and targeting regions, indicating they were separate campaigns. This instance demonstrates how conflicting intelligence signals (i.e. counteracting generation) can destabilize the attribution process and provoke spurious merges or label flipping, consistent with the root cause.

Counteracting generation (2.6) in ❸ prediction. Prediction tasks amplify this vulnerability because they rely heavily on temporal modeling and statistical consistency. Contradictory knowledge (such as

exploitability assessments that are both "confirmed weaponized" and "no evidence of exploitation" across sources) feeds into EPSS-like forecasting or campaign escalation modeling. When these inconsistencies are incorporated into LLM fine-tuning or inference prompts, the model's predictive logic counteracts itself: one reasoning path projects high exploit likelihood, another projects negligible risk. This tension destabilizes alignment objectives that prioritize consistency, leading to incoherent forecasts (e.g., fluctuating risk scores or internally contradictory justifications). Unlike contextualization, where noise merely inflates context, prediction magnifies contradictions because probabilistic reasoning depends on stable and non-conflicting event histories.

Counteracting generation (2.6) in ❹ mitigation. Mitigation is also deeply impacted by counteracting CTI generation and alignment. Conflicting reports about patch effectiveness, bypass proofs-of-concept, or mitigation success create alignment conflicts in LLMs fine-tuned for defensive recommendations. For example, one dataset labels a patch as effective, while another includes verified exploit bypasses; the LLM's training gradients pull in opposite directions, undermining its ability to converge on stable defensive advice. At inference time, this misalignment appears as contradictory recommendations (e.g., suggesting both to apply a patch and to consider it ineffective), undermining trust in automated response playbooks. Summarization models trained on contradictory mitigation corpora may even blend conflicting advice into incoherent instructions. The core issue is that mitigation depends on aligning outputs with actionable truth, but contradictions in defensive knowledge force the LLM into unstable compromise states.

Distributional bias (3.1) in ❶ contextualization. Distributional bias arises most visibly during contextualization because this stage depends heavily on retrieval and enrichment of raw threat data. The corpora used to train retrieval models, event extractors, or knowledge base mappers are often skewed toward particular regions, languages, or reporting sources. For instance, CTI feeds may be dominated by English-language advisories from North American vendors, while reports from smaller regions or niche industry sectors remain underrepresented. As a result, contextualization pipelines learn to prioritize patterns, entities, or CVEs that appear frequently in this skewed distribution, while overlooking less-reported threats. Topic modeling might cluster threats disproportionately around well-documented malware families, and knowledge base mapping might fail to align entities when they originate from underrepresented ecosystems. This distributional imbalance causes models to generalize poorly, enriching raw observations with context that reflects majority patterns but misses critical minority cases (e.g., region-specific campaigns, IoCs from less-visible actors). Thus, contextualization is the primary CTI stage where distributional bias manifests as a root cause of vulnerabilities.

Case Study: Distributional bias (3.1) in ❶ contextualization.

In our CTI contextualization logs, we observed a case tied to **CVE-2023-23397 (Microsoft Outlook elevation of privilege / information disclosure)** in which the enrichment pipeline failed to surface contextual linkage information because the CVE had sparse coverage in English-language reporting. Specifically, although a few localized reports in Eastern European and Southeast Asian languages described exploitation of CVE-2023-23397 with associated infrastructure and campaign details, our retrieval and KB mapping modules did not effectively index or map those non-English sources. Consequently, when processing raw IOC references to that CVE, the system enriched them only with generic Microsoft advisories and common attack campaign metadata, missing region-specific attribution details (e.g. unique malware variants, domain registrants, local threat actor groups). Because the contextualization module overly prioritized cues from the majority (English, widely reported CVEs), it generated weaker context for this vulnerability in our dataset, possibly leading downstream modules to misjudge its significance or misalign attribution/mitigation decisions.

Distributional bias (3.1) in ❷ attribution (Why not influential). Attribution relies less on broad population-level distributions and more on linking observed TTPs, infrastructure, and stylistic features to known actor profiles. While biases in contextualization may already propagate upstream, attribution itself is not directly driven by skewed distributions in the training corpus. Instead, its errors tend to stem from relation extraction and graph construction mistakes, or from contradictory knowledge between sources. Distributional bias is not a first-order effect here, because attribution decisions focus on specific co-occurrence signals (e.g., malware reused by an actor) rather than frequency-based generalizations from an imbalanced corpus.

Distributional bias (3.1) in ❸ prediction (Why not influential). Prediction tasks, including estimating exploit likelihood or forecasting campaign escalation, are typically modeled using temporal trends,

event correlations, or statistical/graph forecasting methods. These processes are less vulnerable to raw data distributional imbalance, since they focus on dynamics over time rather than sheer frequency across corpora. Failures in prediction are more often tied to unseen patterns from emerging threats (3.2) or overfitted reasoning (3.3) rather than distributional skew. Therefore, while contextualization may introduce bias into the upstream data, prediction systems themselves are not intrinsically exposed to distributional bias as a root cause.

Distributional bias (3.1) in ❹ mitigation (Why not influential). Mitigation involves mapping observed vulnerabilities or TTPs to defensive actions, ranking candidate countermeasures, and generating structured recommendations. These outputs are guided by standards (e.g., CVSS), expert-validated mappings, or structured rule sets such as YARA. Because mitigation strategies are less dependent on the global distribution of training data and more on explicit mappings between threats and responses, distributional bias has minimal direct impact here.

Unseen patterns (3.2) in ❶ contextualization (Why not influential). Unseen patterns from emerging threats are less problematic at the contextualization stage. This stage primarily involves collecting raw indicators, mapping them to known entities, and enriching observations with structured taxonomies (e.g., CVEs, MITRE ATT&CK). Since the process focuses on retrieval, normalization, and alignment with established knowledge bases, its outputs remain bounded by what is already recorded in CTI repositories. While contextualization may miss completely novel attack primitives (e.g., a new exploit chain not yet in ATT&CK), it does not typically generate spurious reasoning about unseen patterns, instead, it simply fails to retrieve or map them. Thus, the absence of emerging threat knowledge affects coverage rather than causing misleading generalization errors.

Unseen patterns (3.2) in ❷ attribution. Attribution is more vulnerable to unseen patterns because it requires linking threat behaviors to known adversary profiles. Emerging campaigns may adopt novel TTP combinations, infrastructure setups, or linguistic styles that diverge from historical actor profiles. Relation extraction and graph construction tools, trained on prior threat data, attempt to force these novel behaviors into existing schemas, leading to brittle or incorrect actor assignments. For instance, a new APT might blend techniques previously seen in multiple groups, confusing classifiers that rely on canonical actor–TTP associations. The root cause lies in attribution models' dependence on closed-world assumptions: when novel tactics appear, they are often misclassified into the closest known actor archetype rather than recognized as new.

Case Study: Unseen patterns (3.2) in ❷ attribution.

In our attribution dataset, we observed a campaign exploiting **CVE-2022-30190 (Follina Microsoft Office Remote Code Execution)** whose behavioral signature combined TTPs typical of both Group X and Group Y (e.g. custom VBA macro infection + unusual DNS tunneling), yet also introduced a new lateral movement module not seen before. The attribution model forced the campaign into Group Y because the overlapping macro artifacts and DNS patterns had been heavily associated with Group Y in training. However, deeper analysis of payload internals and command semantics showed the lateral movement logic was markedly different from any known Group Y campaign and the targeting region also diverged. Because the attribution system attempted to "fit" the new pattern into the nearest known actor schema rather than flagging it as novel, the campaign was misattributed. This case illustrates how unseen pattern adoption can lead attribution models to overcommit to the nearest known archetype, thereby misclassifying new or hybrid campaigns.

Unseen patterns (3.2) in ❸ prediction. Prediction tasks are especially sensitive to unseen patterns, since they rely on temporal correlations and trend extrapolation from historical data. Emerging threats often introduce entirely new exploit vectors (e.g., chaining vulnerabilities across cloud microservices) or target previously untapped sectors. Forecasting models built on past timelines cannot anticipate such discontinuities, leading to underestimation of risk or misidentification of targets. For example, EPSS-like scoring frameworks may assign low exploitability probability to a vulnerability because similar CVEs had no known exploitation, only to be proven wrong when a novel exploit technique emerges. Here, the root cause is distributional shift: the statistical regularities captured by time series or graph neural networks no longer hold when threat actors innovate outside historical baselines.

Unseen patterns (3.2) in ❹ mitigation. Unseen patterns also impair mitigation, particularly in the design of defensive playbooks and countermeasure recommendations. When attackers deploy new TTPs or exploit methods absent from training data, mitigation mapping systems may fail to suggest effective countermeasures. For example, an LLM-guided playbook generator may recommend patching or firewall rules aligned with familiar techniques, but overlook mitigations needed for an

entirely new lateral movement strategy. Similarly, mitigation efficacy predictors struggle because they assume the space of attack vectors is known and represented in past cases. The root cause here is defensive brittleness: mitigation frameworks generalize from established mappings, and emerging patterns invalidate those assumptions, leading to incomplete or misaligned recommendations.

Overfitted reasoning (3.3) in ❶ contextualization (Why not influential). Overfitted reasoning is less relevant in contextualization because this stage primarily focuses on enriching raw observations with metadata and aligning them to structured identifiers. Techniques like topic modeling or information retrieval operate on co-occurrence and similarity rather than predictive inference, meaning they are less prone to the memorization-driven brittleness characteristic of overfitting. Errors at this stage are more often due to spurious correlations or semantic conflicts, not reinforcement of memorized reasoning paths.

Overfitted reasoning (3.3) in ❷ attribution. Overfitted reasoning becomes prominent in attribution when models repeatedly learn shallow associations between specific indicators and adversary labels. For instance, if relation extraction and entity linking pipelines are disproportionately trained on a limited set of well-documented campaigns, the system may memorize that certain infrastructure patterns (e.g., recurring domains or malware family strings) always belong to a specific actor profile. When new reports mention similar but unrelated infrastructure, the model may reflexively attribute them to the memorized actor without properly considering alternative explanations. This overreliance on memorized co-occurrences results from insufficient exposure to diverse attribution cases, leading to brittle classification of actors and campaigns.

Case Study: Overfitted reasoning (3.3) in ❷ attribution.

In our attribution result logs, we discovered a case linked to **CVE-2020-1472 (Netlogon Elevation of Privilege Vulnerability, aka Zerologon)** where the attribution model incorrectly assigned a campaign to "Actor Alpha" purely because of reused domain naming conventions and IP subnets that had earlier been heavily associated with Actor Alpha in training data. In reality, forensic investigation showed that the campaign used distinct command-and-control servers, payload variants, and targeting patterns, inconsistent with Actor Alpha's known modus operandi. Because the model had overly internalized the co-occurrence of those domains and subnets with Actor Alpha in its limited training corpus, it defaulted to attributing new instances to that actor without hypothesizing alternate actors or considering the evidence diversity. This mismatch exposes how overfitted reasoning can lead to misattribution in real-world CTI pipelines.

Overfitted reasoning (3.3) in ❸ prediction. In prediction, overfitted reasoning manifests when forecasting models generalize poorly beyond historical data. Temporal modeling and event correlation techniques often capture strong patterns within past campaigns, such as the escalation of a vulnerability class into active exploitation. However, if these forecasting models are overly tuned to such repeated sequences, they may incorrectly predict the same escalation dynamics for future, unrelated vulnerabilities. For example, the model might overestimate exploitation probability simply because prior vulnerabilities of a similar type were exploited, even though current conditions differ. This bias reflects overfitting to observed sequences, where models memorize recurring attack trajectories instead of reasoning about underlying causal drivers of exploitation.

Overfitted reasoning (3.3) in ❹ mitigation (Why not influential). Mitigation tasks also exhibit limited vulnerability to overfitted reasoning. While mitigation mapping and efficacy ranking involve inference, they generally rely on explicit rule associations or empirical evaluations of patch effectiveness. Summarization and playbook generation are shaped by aggregation of defensive knowledge rather than predictive modeling of future adversarial behavior. Consequently, errors in this stage stem more from contradictions in data sources or co-mention bias in countermeasure lists than from overfitting to historical reasoning trajectories.

Environmental unawareness (3.4) in ❶ contextualization (Why not influential). Environmental unawareness is less prominent during contextualization because this stage focuses on gathering and aligning observable facts (e.g., IOCs, vulnerability references, or malware labels) rather than reasoning about the environment where attacks unfold. The techniques employed (topic modeling, event extraction, knowledge base mapping, retrieval) primarily enrich raw data without needing to adapt to host- or sector-specific conditions. Since contextualization tasks are mostly descriptive and taxonomy-driven, the absence of local system or organizational environment data does not strongly distort their outputs. As a result, contextualization is not significantly affected by this type of vulnerability.

Environmental unawareness (3.4) in ❷ attribution. In attribution, environmental unawareness manifests when models ignore the operational or deployment context in which infrastructure is reused. For instance, relation extraction or event graph construction may connect domains, malware families, or command-and-control servers across multiple incidents, but fail to recognize that one set of infrastructure belongs to a staging environment while another is tied to production systems in a different sector. Without environmental cues, models over-attribute incidents to the same actor or campaign, producing inflated linkages. Similarly, behavioral classification often overlooks local defender responses or system baselines that would otherwise clarify whether repeated TTPs reflect adversary persistence or benign background activity. The root cause is that attribution pipelines assume global uniformity of threat behavior while overlooking environment-specific nuances that separate true operational reuse from coincidental overlap.

Environmental unawareness (3.4) in ❸ prediction. In prediction tasks, environmental unawareness becomes more pronounced because forecasting inherently requires understanding the conditions under which threats evolve. Temporal modeling or graph neural networks may detect sequences of exploits but fail to adjust for environmental variables such as patch adoption rates, geographic regulatory differences, or sector-specific exposure. As a result, a vulnerability exploited in one industry may be wrongly forecast as high-risk for another, even though the latter has stronger baseline defenses or different system architectures. Similarly, probability estimates for exploitation (e.g., EPSS-like scoring) may ignore localized security controls or asset configurations, leading to overly broad or inaccurate risk forecasts. The root cause is the assumption that historical global patterns can be applied uniformly, when in fact exploitability is mediated by local system and organizational environments.

Environmental unawareness (3.4) in ❹ mitigation. In mitigation, environmental unawareness leads to defensive strategies that are technically correct in general but ineffective in practice for specific deployments. For instance, mitigation mapping may recommend a patch that is incompatible with legacy systems, or propose firewall rules that disrupt legitimate sector-specific workflows. Mitigation efficacy prediction models often rank countermeasures without considering resource constraints, organizational processes, or compliance requirements, resulting in impractical prioritization. Summarization modules may produce generic remediation steps that fail to address custom software stacks or hybrid cloud deployments. The root cause is the lack of integration between CTI outputs and real-world operational contexts, causing recommended actions to miss alignment with the defender's environment, and ultimately weakening the utility of CTI-driven defense.

Case Study: Environmental unawareness (3.4) in ❹ mitigation.

In our CTI-backed vulnerability-response dataset, we identified a representative instance involving **CVE-2021-44228 (Log4Shell)** where mitigation suggestions failed to account for environmental constraints. Specifically, a model-generated recommendation ranked upgrading to the latest Log4j version (2.16 or above) as the top-priority action. However, in certain enterprise deployments, this upgrade conflicted with custom-built logging plugins and legacy compatibility modules, resulting in logging failures and application crashes. Due to concerns over business continuity, the organization delayed patch deployment despite the known severity of the vulnerability. Additionally, the model suggested firewall rule updates to restrict inbound JNDI traffic, which inadvertently disrupted legitimate cross-tenant log aggregation workflows in a hybrid cloud environment. These misalignments between the recommended actions and the operational realities led to non-adoption of the mitigation plan, illustrating how environmental unawareness can undermine CTI-guided defense.

G.4 Additional Analyses of Intertwined Failure Modes

In this subsection, we provide extended analyses to deepen our understanding of intertwined failure modes in CTI modeling. We focus on three complementary aspects: (i) the sequential accumulation of failures across CTI stages, (ii) the concurrent presence of multiple failure modes in the threat landscape, and (iii) detailed case studies that highlight how intertwined failures manifest in practice.

Accumulation across CTI stages. The CTI pipeline is inherently sequential, with outputs from early modules serving as inputs for downstream reasoning. When an upstream stage introduces an error, such as co-mention bias in event contextualization, this misinformation propagates forward as if it were ground truth. In attribution, the model may then reinforce the biased linkage (e.g., mapping a benign domain to a threat actor), while in prediction, it may extrapolate incorrect exploitability trends based on the faulty assumption. Similarly, skewed source reliance during retrieval can lock later

stages into one-sided perspectives, preventing correction even when contradictory evidence emerges. Over time, these sequentially inherited errors accumulate into cascades, where a single misstep at the contextualization stage magnifies into systemic reasoning failures across attribution, prediction, and mitigation.

Concurrent presence in the threat landscape. Beyond sequential propagation, failure modes also co-occur within the same analytical slice of a threat landscape. For example, constrained generalization failures often combine: a distributional bias (e.g., defaulting to ransomware explanations) may overlap with environmental unawareness (e.g., ignoring that the attack only targets Linux servers). Likewise, unseen patterns from emerging threats frequently intersect with overfitted reasoning, as the model forces novel evidence into familiar but inaccurate templates. These concurrent failure modes are not merely additive but entangled, since the existence of one (e.g., mislabeling the environment) amplifies the harm of another (e.g., failure to adapt to an unseen pattern). This reflects the reality of CTI data, where heterogeneous and incomplete sources naturally produce overlapping inconsistencies.

Case Study: Cloud API exploitation (2024 campaign).

Reports diverged semantically and temporally, with some describing privilege escalation via API tokens and others treating it as lateral movement. The model conflated these into a hybrid description (semantic conflict), while also failing to recognize the pattern as novel (unseen pattern). Environmental unawareness compounded the issue when the output generalized the vulnerability to all cloud services, despite references limiting it to a specific provider.

These analyses show that intertwined failure modes are not isolated anomalies but systemic consequences of how CTI evidence is structured, consumed, and reasoned upon by LLM-based agents. Recognizing their cumulative and concurrent nature is essential for designing defenses that target the compounding rather than the individual failure.

H Complementing Details of Defensive Strategy

Based on the experimental insights (§4), we further counteract CTI failure modes at two levels of granularity: (1) a universal intervention applied during inference-time LLM reasoning to defend against all potential failure modes, and (2) a set of fine-grained, qualitative mitigations applied to cyberspace.

H.1 A causal intervention design to validate and mitigate CTI Failure Modes

Design principle: counterfactual consistency. Despite significant differences, the three failure mode classes (spurious correlation, contradictory knowledge, and constrained generalization) share a common failure pattern in LLM reasoning: models tend to overfit to surface-level statistical co-occurrences rather than the *invariant causal attributes* that actually determine the correct CTI judgment.

Definition ✎: Causal attributes refer to CTI factors that should causally drive the label or conclusion, e.g., the attacker's core *TTPs* (techniques, procedures, and behavioral signatures), *exploitation mechanism*, *malware or tool behaviors/capabilities*, and other stable indicators of the underlying attack process, rather than incidental context such as timestamps, victim industry, geography, or reporting-source artifacts.

To mitigate CTI failure modes, we propose an inference-time **counterfactual consistency check** (C^3) based on a simple principle: if a CTI judgment is truly driven by these causal attributes, it should remain stable when we perturb *non-causal* contextual tokens (e.g., dates, sectors) that should not affect the conclusion. Formally, we model CTI analysis as a mapping $f(x) \to y$. We then define a set of targeted intervention operators $\mathcal{I} = \{I_{\text{spur}}, I_{\text{contra}}, I_{\text{gen}}\}$ that generate counterfactual variants x'_k by editing the non-causal tokens associated with each failure type, while preserving the CTI context (i.e., the causal attributes) that is leveraged to determine y.

Validation (diagnosis). To validate the existence of latent failure modes without ground truth labels, we utilize C^3 as a sensitivity-based detector. For a given input x and a target prediction y, we compute

Table 10: Absolute improvement (Δ) in CTI task performance following causal interventions. The proposed method targets all three failure types (spurious correlation, contradictory knowledge, and constrained Generalization) at inference time reasoning. Each number denotes the raw gain (e.g., F1 + Δ) over the original performance (Table 2).

Detailed CTI Task	Metric	Industry-leading, general-purpose						Cybersecurity-specialized					
		G5	CLD	GEM	LL70	MIX	QWN	FSC	ZYS	LLY	CBS	SPT	DHT
❶ Contextualization													
Affected Systems	F1	3.1	5.5	6.8	2.2	6.1	5.4	3.8	3.5	3.3	3.9	4.2	3.8
Attack Infrastructure	F1	2.5	6.2	7.1	3.1	6.8	6.5	5.2	4.1	3.8	4.5	4.8	4.4
Vulnerability Linking	Acc	4.2	6.8	7.5	4.8	7.2	6.8	5.8	3.9	3.5	4.1	4.4	3.9
Malware Family Mapping	F1	4.5	7.1	8.2	5.5	7.9	7.4	6.2	4.8	4.2	5.1	5.5	4.9
IOC Normalization	F1	1.8	3.5	4.1	2.2	3.8	3.5	2.9	2.1	1.8	2.5	2.8	2.2
Threat Report Alignment	BLEU	8.2	11.2	9.8	7.5	10.5	9.8	9.1	6.2	5.8	7.5	8.8	6.5
Event Timeline Construction	BLEU	7.5	9.5	9.1	8.2	9.4	8.9	8.5	5.5	4.8	5.8	6.2	5.5
Graph Population	Acc	3.5	7.5	8.1	4.2	8.5	7.2	6.8	4.5	4.2	3.8	6.5	3.5
Source Reliability Scoring	AUC	1.2	2.8	2.1	1.8	3.2	2.5	2.5	1.5	1.2	1.8	2.2	1.5
❷ Attribution													
Threat Actor Linking	Acc	2.1	5.5	4.2	3.5	5.2	4.8	5.8	2.5	1.8	2.8	2.5	3.5
TTP Extraction	F1	3.2	4.8	5.2	4.5	5.5	5.1	4.2	3.1	2.8	3.5	3.8	3.2
Campaign Attribution	Acc	4.8	6.8	7.2	5.2	7.5	7.1	5.8	4.2	3.8	4.5	4.8	4.2
Infrastructure Reuse	F1	5.2	7.5	8.1	5.8	8.2	7.8	6.5	4.5	4.1	4.8	5.2	4.5
Language/Style Profiling	Acc	2.5	4.5	4.8	3.8	5.2	4.8	4.1	2.8	2.5	3.2	3.5	2.8
False Flag Detection	F1	8.8	10.8	11.2	8.2	11.5	10.5	9.5	7.5	6.8	8.2	8.5	7.8
Evidence Weighting	BLEU	9.2	12.5	11.5	8.8	11.8	11.2	9.8	7.2	6.5	8.5	9.2	7.5
Relation Graph Building	F1	4.2	6.2	6.8	4.8	6.5	6.1	5.2	3.8	3.5	4.2	4.5	3.8
❸ Prediction													
Exploit Likelihood	AUC	1.5	3.5	3.8	1.8	4.2	3.8	4.5	2.2	1.8	1.5	2.5	1.8
Impact Forecast	BLEU	6.5	9.2	8.5	5.8	9.5	8.8	7.5	5.2	4.8	5.8	6.5	5.2
Target Sector Prediction	Acc	2.8	5.2	4.5	3.2	5.5	5.1	4.8	3.5	3.1	2.8	3.8	3.2
Campaign Escalation	AUC	4.5	7.2	7.8	5.2	7.5	7.1	6.2	4.8	4.5	5.2	5.5	4.8
❹ Mitigation													
Patch Recommendation	F1	3.8	5.8	6.2	3.2	6.5	6.1	5.2	4.1	3.8	3.2	3.5	4.5
Rule Generation (YARA)	BLEU	5.2	8.2	7.5	5.8	8.5	7.8	6.8	4.5	4.2	5.2	5.5	4.8
Response Summarization	BLEU	4.2	6.8	6.2	4.8	7.2	6.5	5.5	3.8	3.5	3.2	4.5	3.8
Mitigation–TTP Mapping	Acc	3.5	5.5	5.8	4.1	6.2	5.8	4.8	3.2	2.8	3.1	3.5	3.2
Defensive Playbook Gen	BLEU	6.2	8.8	8.2	6.8	9.2	8.5	7.2	5.5	5.1	5.2	5.8	5.5
Countermeasure Ranking	NDCG	5.5	8.2	7.8	6.1	8.5	7.8	6.5	4.8	4.5	4.2	5.5	5.8
Incident Ticket Generation	Acc	2.5	4.5	3.8	1.8	4.8	4.5	3.8	2.8	2.5	2.2	2.8	2.5

a *Consistency Score* Ω:

$$\Omega(y|x) = \frac{1}{|\mathcal{K}|} \sum_{k=1}^{|\mathcal{K}|} \mathbb{1}\left(\text{argmax}_{y'} P(y'|I_k(x)) = y\right) \tag{3}$$

where $\mathcal{K}$ is the manifold of generated counterfactuals. If $\Omega(y|x) < \tau$ (a stability threshold), the prediction is flagged as "`Vulnerable`." For example, if changing the report's date causes the model to flip its attribution from an APT identifier to "unknown," we validate the presence of a *contradictory knowledge* failure (Type 2.1). This metric provides a quantifiable proxy for model robustness, allowing us to detect specific failure modes in real-time.

Inference-time mitigation (correction). The C^3 framework goes beyond diagnosis and can serve as an active mitigation engine. Instead of simply relying on the greedy decoding of the original input, we compute the final output $\hat{y}$ via a weighted consensus across the counterfactual set:

$$\hat{y} = \text{argmax}_y \sum_k w_k \cdot P(y|x'_k) \tag{4}$$

where w_k penalizes paths with high entropy. This effectively filters out reasoning driven by spuriousness (Type 1.x), contradictions (Type 2.x), or overfitting (Type 3.x) by forcing LLM reasoning to converge on evidence that is invariant across contexts.

Experimental results. We empirically show the efficacy of this design by applying the C^3 intervention across our benchmark tasks. Table 10 details the absolute performance gains (Δ) observed when switching from standard inference to our causal mitigation. We observe a universal improvement, particularly in tasks sensitive to high-noise metadata. For instance, *false flag detection* presents a significant improvement (up to $+0.115$ F1 for Mixtral), demonstrating that spurious correlation was indeed the primary bottleneck. Similarly, *event timeline construction* improves significantly across all

Table 11: Ratio of failure modes (same as Figure 1) after applying tailored mitigations. Lower ratio is better.

Method		Spurious Correlation					Contradictory Knowledge						Constrained Generalization			
		1.1 Co-men. bias	**1.2** Exploit. bias	**1.3** Confond. factor	**1.4** Skewed source	**1.5** Hier. metadata	**2.1** Temporal contra.	**2.2** Conflict report	**2.3** Semantic conflict	**2.4** Diver. data	**2.5** Misal. knowl.	**2.6** Counter. CTI	**3.1** Distribut. bias	**3.2** Unseen pattern	**3.3** Overfit. reasoning	**3.4** Env. unaware
General	Retrieval-only	3.12	2.05	10.45	2.15	2.11	4.25	6.82	3.55	2.12	1.15	9.45	18.45	19.56	2.55	3.12
	Inference-only	1.85	1.15	5.65	5.23	1.89	10.45	14.56	2.89	3.15	1.95	5.12	12.34	11.45	1.12	5.45
	Full-fledged	1.05	0.65	3.12	1.85	0.95	2.55	4.15	1.25	1.55	0.65	3.25	9.56	7.85	0.45	2.15
Agent	Retrieval-only	2.56	0.72	13.45	1.12	0.55	2.15	5.12	1.15	1.95	0.45	1.89	19.56	24.12	16.45	6.89
	Inference-only	1.25	0.35	7.85	3.45	0.48	6.89	10.45	0.85	3.12	0.65	0.85	13.45	14.56	8.45	11.25
	Full-fledged	0.85	0.22	4.56	0.95	0.25	1.55	3.45	0.45	1.25	0.25	0.45	10.56	11.25	5.12	5.45

models (avg $\Delta \approx 0.08$), showing that C^3 successfully mitigates *contradictory knowledge* conflicts. Notably, while specialized agentic models show smaller relative gains due to their higher baseline, the intervention still yields consistent improvements, proving that these causal failure modes are inherent to the LLM architecture rather than just domain knowledge gaps.

H.2 Fine-grained mitigation tailored to specific failure mode subtypes

Building on the causal intervention results above, we now design a set of fine-grained, actionable mitigation strategies tailored to each failure mode subtype.

H.2.1 Mitigating spurious correlation failures

Spurious correlation failures arise when LLMs overgeneralize from patterns that are statistically associated but not causally linked. These failures typically stem from co-occurrence signals, reused observables, or implicit biases embedded in CTI report structures and metadata. To mitigate such issues, we recommend a combination of **(i) retrieval-time** evidence filtering and **(ii) prompt-level** constraints, which incorporate causal reasoning and uncertainty modeling to reduce the reliance on misleading associations and encourage evidence-based reasoning. Below, we detail tailored designs:

1.1 Co-mention bias from raw threat incident. *(i) Retrieval:* run entity resolution + role labeling, then keep only entities with *incident-role evidence* (e.g., "operator," "infrastructure owner," "victim") and down-rank "mentioned-in-passing" names. *(ii) Inference:* require an *evidence link* for every predicted actor/tool (quote/span + relation type), and add a simple rule: *reject if the only support is co-mention*.

1.2 Exploitation bias from deliberately reused IoCs. *(i) Retrieval:* maintain an IoC reuse profile (first/last seen, frequency, and how many different campaigns it appears in) and down-rank "high-reuse" IoCs unless they come with *campaign-specific* behavioral clues (delivery method, exploit chain, payload traits). *(ii) Inference:* force a two-step check: first ask "is this IoC discriminative here, or commonly reused?", then only make attribution/recommendations if it is discriminative.

1.3 Confounding factors that explicitly/implicitly correlate entities. *(i) Retrieval:* separate *causal evidence* (behavior, exploit mechanism, toolmarks, infrastructure ownership) from *incidental context* (sector, region, time), and limit how much context-only metadata can drive retrieval ranking. *(ii) Inference:* add a counterfactual prompt: "would the conclusion change if we changed sector/region/date?" If yes, lower confidence or abstain, and ask for causal evidence.

1.4 Skewed source. *(i) Retrieval:* enforce source diversity (e.g., at least k different publishers) and de-duplicate near-identical reports; attach source + timestamp tags to each claim. *(ii) Inference:* require multi-source support for high-confidence claims; otherwise summarize the disagreement ("A says X, B says Y") and output a ranked hypothesis list with uncertainty.

1.5 Hierarchical metadata from attack chains. *(i) Retrieval:* first treat kill-chain stages as *soft* hints unless backed by artifacts (telemetry/logs); then attach a confidence note to each stage transition. *(ii) Inference:* add a strict rule: do not infer later-stage actions unless there is explicit evidence for them. If the chain contains incomplete evidence, keep the answer partial on present metadata.

H.2.2 MITIGATING CONTRADICTORY KNOWLEDGE FAILURES

Contradictions occur when models encounter conflicting, inconsistent, or incompatible CTI between different threat databases or platforms. These contradictions can arise from differences in temporal scope, analyst interpretation, data formats, or semantic misalignment. To mitigate these failures, systems should incorporate **(i) retrieval-based** temporal or source-aware annotation and **(ii) inference-time** conflict detection and uncertainty justification. Detailed strategies are:

2.1 Temporal contradiction between outdated and recent evidence. *(i) Retrieval:* time-aware reranking (prefer recent + corroborated sources), and package evidence in chronological order with clear `as-of` time tags. *(ii) Inference:* when claims conflict, prefer newer evidence unless it is clearly weaker; otherwise present both with timestamps and explain the reason of conflict in a self-reflective generation.

2.2 Conflicting reports of attack contexts or dependencies. *(i) Retrieval:* cluster retrieved passages into consistent views (by source or by claim similarity) and feed them as grouped bundles rather than a mixed list. *(ii) Inference:* require a short divergence summary (e.g., which vendor and information source claim what scenarios) and a decision policy: choose one with justification, or abstain and list what extra evidence would resolve it.

2.3 Semantic conflict. *(i) Retrieval:* canonicalize entities (aliases, malware family names, tool versions) using an alias table/KB, and attach disambiguation notes when names are similar. *(ii) Inference:* add a gate before merging claims: "same entity, alias, or related-but-different?" and do not merge unless it is truly the same; also explicitly check for negation (e.g., not attributed).

2.4 Divergent data structure from different platforms. *(i) Retrieval:* normalize different schemas (e.g., STIX, vendor formats) into one simple intermediate record (typed fields or triples) before passing to the model. *(ii) Inference:* feed the model the normalized record with field definitions, and instruct it to reason from field-specific meaning (prompt derived from schemas).

2.5 Misaligned knowledge and security standards. *(i) Retrieval:* attach authoritative references during retrieval (valid MITRE ATT&CK technique IDs, CVSS rubric snippets, NVD-style conventions) and validate candidate mappings early. *(ii) Inference:* constrain outputs to valid options (pick from a provided list); if evidence does not support a clean mapping, output "unknown/insufficient evidence" with a brief rationale.

2.6 Counteracting CTI generation and LLM alignment. *(i) Retrieval:* require every major claim to carry a grounding span (quote + source + time), and keep an audit trail of which evidence supported which claim. *(ii) Inference:* enforce concise bullet claims with a citation to evidence, plus calibrated uncertainty (no confident wording without support).

H.2.3 MITIGATING CONSTRAINED GENERALIZATION FAILURES

Generalization issues in cybersecurity often present as insufficient understanding on emerging threats or adversarial shifts in attack behavior (Nam, 2019). To mitigate these issues, systems should incorporate **(i) retrieval-based** data augmentation to collect auxiliary knowledge from external sources and **(ii) inference-time** generalization (rather than memorization) to emphasize task-specific reasoning:

3.1 Distributional bias. *(i) Retrieval:* balance CVE exposures by reweighting retrieval results so long-tail actors/regions/tools are not drowned out; add simple quotas or diversity constraints by geography/tooling when possible. *(ii) Inference:* explicitly warn against common threat actors (or infrastructure) and require the deliberation to cite causal indicators (behavior/exploit/toolmarks), not just region/sector stereotypes.

3.2 Unseen pattern from emerging threats. *(i) Retrieval:* plug in fresh CTI feeds (recent reports, disclosures, malware analysis notes) with freshness filters and credibility scoring, so the context reflects post-training threats. *(ii) Inference:* use a novelty-aware response: propose cautious hypotheses, clearly mark what is unknown, and list the top evidence signals that would confirm or refute each hypothesis.

3.3 Overfitted reasoning. *(i) Retrieval:* diversify evidence so one shortcut (e.g., tool $\rightarrow$ actor) cannot dominate; when possible, retrieve counterevidence or alternatives. *(ii) Inference:* add an

evidence-sufficiency gate: if attribution is not explicitly supported, lower confidence or abstain; optionally run a small counterfactual check that swaps common shortcuts (actor/infra) to see if the conclusion collapses.

3.4 Environmental unawareness. *(i) Retrieval:* treat environment details as primary inputs (including but not limited to asset type, deployment, sector constraints, region) and retrieve environment-matched guidance and prior cases. *(ii) Inference:* require conditional reasoning: "given environment E, risk/mitigation changes because…", and tie incident response recommendations to environment-specific constraints.

H.2.4 EXPERIMENT RESULTS ON MITIGATION

The experimental results in Table 11 demonstrate that the above approaches yield consistent reduction in failure mode rates across all categories, outperforming both ablation baselines (i.e., retrieval-only or inference-only mitigation). A comparative analysis of the ablations reveals distinct component strengths. Retrieval-only mitigation proves superior in resolving contradictory knowledge (Types 2.1, 2.2, 2.4) and environmental unawareness (Type 3.4). This suggests that failures driven by outdated information or source ambiguity are best addressed by grounding the model in retrieved, verifiable evidence (e.g., reducing General-LLM temporal contradiction from 17.53% to 4.25%). This improvement implies that external evidence serves as an auxiliary signal for time-sensitive or source-conflicting queries.

Conversely, Inference-only mitigation dominates in mitigating spurious correlations (Types 1.1, 1.3) and constrained generalization (Types 3.2, 3.3). For instance, in *confounding factors* (1.3), the inference-based check reduces the failure rate for General LLMs to 5.65% (vs. 10.45% for retrieval), indicating that these errors stem from flawed internal reasoning rather than a lack of information. The full-fledged mitigation further synthesizes retrieval to cleanse the input context and causal inference to filter the reasoning process, thereby achieving the lowest universal failure ratios.