BENCHMARK SCORES ARE PIPELINE-DEPENDENT: A RELIABILITY AUDIT OF CYBERSECURITY LLM BENCHMARKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) benchmarks are often treated as fixed datasets with stable scores, yet their outcomes depend on configurable evaluation pipelines. We audit eight cybersecurity benchmarks across 10 proprietary, open-weight, and cybersecurity-specialized LLMs. By modeling benchmarks as measurement pipelines, we identify 15 systematic failure modes and show that a single pipeline choice can change a model's score by more than 80 percentage points and substantially alter model rankings. At the cross-benchmark level, two semantically similar task pairs rank the same models differently because of incompatible evaluation conventions. Under an evaluation harness that standardizes pipeline choices while preserving task semantics, nine of 10 models shift by at least three ranks on at least one benchmark. These results show that cybersecurity LLM benchmark scores are pipeline-dependent and motivate pipeline-aware auditing as a core requirement for reliable model evaluation.

1 INTRODUCTION

Large language model (LLM) evaluation is increasingly benchmark-driven. Benchmark scores guide model selection, support claims of state-of-the-art performance, influence deployment decisions, and shape leaderboards. Yet, these scores are often interpreted as stable measurements of model capability, even though they are produced by configurable evaluation scripts involving prompts, inference settings, output extraction, scoring, and aggregation. In generative settings, where outputs are open-ended and extraction is heuristic, benchmark scores can reflect the extraction step as much as model capability (Shi et al., 2024; Sun et al., 2024; Wang et al., 2023).

We study this problem in cybersecurity, a high-stakes domain whose benchmarks span factual recall, vulnerability analysis, threat-intelligence extraction, attacker attribution, attack-technique mapping, mitigation selection, and general security reasoning. We focus on benchmarks that evaluate knowledge and analytical cybersecurity tasks, and exclude benchmarks that evaluate offensive or defensive capabilities. These tasks depend on evolving, structured authoritative sources such as CVE, CWE, CVSS, and MITRE ATT&CK (Sikos, 2023), making evaluation especially sensitive to pipeline errors and inconsistent labels. Such failures can distort claims of model specialization, obscure genuine capability differences, and mislead deployment decisions. Cybersecurity also provides a mature and heterogeneous benchmark ecosystem that has not been systematically audited, including in recent large-scale meta-evaluations (Bean et al., 2025).

In this paper, we ask: *to what extent do design choices of evaluation pipelines affect the reliability of cybersecurity LLM benchmark outcomes?* We argue that benchmarks should be viewed not as static datasets paired with fixed metrics, but as *measurement pipelines* that transform tasks and model outputs into numerical performance estimates (§2). Under this formulation, a benchmark score is conditional on its evaluation pipeline's stages, consisting of dataset construction, prompt specification, inference, extraction and scoring, and aggregation.

We audit eight cybersecurity benchmarks comprising 48,662 questions across 23 tasks against 10 proprietary, open-weight, and cybersecurity-specialized LLMs (§3) and identify 15 failure modes, observed in 34 benchmark–failure-mode interactions across the evaluation pipeline (§4; Table 1). Fixing these failures can shift a model's benchmark score by over 80 percentage points. For example,

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.

Table 1: Failure-mode incidence across the eight audited benchmarks (red check: observed; green dot: not observed). All $8 \times 15 = 120$ benchmark–failure-mode pairs were examined, yielding 34 observed incidents. Within each pipeline stage, $\mathcal{F}_i$ denotes the i-th failure mode defined in Table 4.

	Dataset		Prompt			Inference			Extraction				Aggregation			
Benchmark	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_4$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	**Total**
MMLU-CS	✓	✓	·	·	·	·	·	·	·	·	·	·	✓	·	·	3
SecEval	·	✓	·	✓	·	·	✓	·	·	·	·	·	·	·	✓	4
SECURE	·	✓	·	·	✓	·	·	·	·	✓	·	·	·	·	·	3
CTI-Bench	·	✓	·	·	✓	✓	·	·	✓	✓	✓	·	·	✓	✓	8
AthenaBench	✓	✓	·	·	·	·	·	·	·	✓	✓	✓	·	✓	·	6
CyberMetric	✓	·	✓	·	·	·	·	✓	·	·	·	·	·	·	·	3
RedSage-Bench	✓	✓	·	·	·	✓	·	·	✓	·	·	·	✓	·	·	5
SecBench	✓	✓	·	·	·	·	·	·	·	·	·	·	·	·	·	2
Total	5	7	1	1	2	2	1	1	2	3	2	1	2	2	2	34

RedSage-Bench (Suryanto et al., 2026) uses "\n" as a stop sequence, causing generation to halt at the first newline. For Qwen3.6, this sequence fires inside the reasoning preamble before any answer token is produced, yielding empty outputs. Generating until the end-of-sequence token, using a token budget large enough for the reasoning span to close, and stripping the reasoning span before extraction, recovers 85.9 percentage points.

Across benchmarks, broader task coverage can still yield redundant measurements and unstable model rankings (§5). Using Principal Component Analysis (PCA), we find that the first component explains 95.25% of the variance across the 23 task $\times$ 10 model score matrix, showing that most tasks largely capture the same broad performance dimension. However, task-induced rankings can still disagree. Two semantically similar task pairs in CTI-Bench (Alam et al., 2024) and AthenaBench (Alam et al., 2025) show weak rank agreement under Kendall's τ-b (0.29 and 0.24), with pairwise inversions revealing changes in specific model orderings. These discrepancies stem largely from incompatible scoring conventions, specifically binary versus partial-credit scoring and alias handling.

To isolate pipeline effects, we implement a harness that standardizes nine pipeline configuration fields, including prompt formatting, decoding, and extraction rules, wherever benchmark semantics permit. Under this harness, nine of 10 models shift by at least three ranks on at least one benchmark (§5). This has an important practical implication: cybersecurity-specialized models are often selected based on leaderboard rankings, yet those rankings can reflect evaluation-pipeline choices as much as underlying domain capability. We use these findings to formulate a set of recommendations for more reliable benchmarking (§6).

Our contributions are threefold. First, we formalize LLM benchmarks as measurement pipelines and audit every pipeline stage of eight cybersecurity benchmarks across 10 LLMs under a common framework. Second, we quantify stage-level effects through controlled perturbations of pipeline configuration fields, showing that pipeline choices affect both model scores and rankings. Third, we release an evaluation harness[1] and audit code[2] that make heterogeneous benchmark assumptions explicit and reproducible, and publish our results on EveryEvalEver (Batzner et al., 2026).

2 Benchmarks as Measurement Pipelines

LLM benchmarks are often described as datasets paired with metrics, where a metric specifies how model performance is quantified and a score is the resulting numerical measurement. In practice, that score is produced by a multi-stage evaluation procedure in which each stage transforms the output of the preceding stage and introduces choices that can affect the final measurement. We therefore model a benchmark as a *measurement pipeline*, represented as a nested composition of stage-specific functions. For benchmark b and model m, the reported score $\mathcal{S}_b(m)$ can be written as:

$$\mathcal{S}_b(m) = \mathcal{A}_b(\mathcal{E}_b(\mathcal{I}_b(m, \mathcal{P}_b(\mathcal{D}_b)))) \,, \tag{1}$$

where $\mathcal{D}_b$ is the benchmark dataset, $\mathcal{P}_b$ the prompt specification, $\mathcal{I}_b$ the inference procedure, $\mathcal{E}_b$ the extraction and scoring procedure, and $\mathcal{A}_b$ the aggregation rule. Under this formulation, the benchmark

[1] https://anonymous.4open.science/r/cyeval-DF7B/
[2] https://anonymous.4open.science/r/audit-F356/

score is the output of the composed evaluation pipeline rather than an intrinsic property of the model alone. As a result, $\mathcal{S}_b(m)$ is conditional on the full pipeline used to produce it.

Pipeline stages. *Dataset construction ($\mathcal{D}_b$)* defines the evaluation distribution, including task coverage, label quality, and answer representation. *Prompt specification ($\mathcal{P}_b$)* determines how tasks are presented to the model, including instructions, demonstrations, chat formatting, and output-format constraints. *Inference configuration ($\mathcal{I}_b$)* controls how responses are generated, including decoding parameters, token budgets, stop sequences, serving backends, and backend-specific constraints. *Extraction and scoring ($\mathcal{E}_b$)* converts model outputs into predictions or graded judgments. *Aggregation ($\mathcal{A}_b$)* combines question- and task-level measurements into final benchmark scores.

The five stages in Eq. 1 are instantiated through nine *pipeline configuration fields*, which we specify, perturb, and standardize throughout the paper: prompt template and chat formatting ($\mathcal{P}$); decoding, maximum new tokens, and stop sequences ($\mathcal{I}$); extraction rule, scoring rule, and denominator policy ($\mathcal{E}$); and aggregation rule ($\mathcal{A}$). App. D.4 records these fields for every benchmark.

2.1 Meta-Evaluation Methodology

The pipeline view separates benchmark reliability into two levels. At the *benchmark level*, we inspect each pipeline for stages where a design or implementation choice changes the reported score. We call such a pattern a *failure mode*: a recurring feature of the evaluation procedure that can change reported scores without any corresponding change in model capability. At the *cross-benchmark level*, we ask whether the audited benchmarks support stable conclusions about which model performs better. We denote the i-th failure mode at pipeline stage $\mathcal{X}_b \in \{\mathcal{D}_b, \mathcal{P}_b, \mathcal{I}_b, \mathcal{E}_b, \mathcal{A}_b\}$ for benchmark b and model m as $\mathcal{F}_i(m, \mathcal{X}_b)$. When discussing failure modes independent of a particular benchmark or model, we use the shorthand $\mathcal{F}_i(\mathcal{X})$ to denote the i-th failure mode at stage $\mathcal{X}$.

We audit every stage of each benchmark, reconstructing the documented and released pipelines independently, recording where they disagree, and identifying choices we must supply because neither specifies them. Where a single pipeline configuration field can be varied while holding the others fixed, we perturb it and measure the change; App. F.1 records where such isolation is not possible. Because not all failure modes admit a per-model score shift, we quantify each with a measure suited to its mechanism (App. F.6).

Each failure-mode analysis is either a *re-scoring* analysis, in which we reuse stored model outputs while changing a downstream pipeline configuration field (e.g., the extraction rule), or a *re-generation* analysis, in which we produce new outputs after changing an upstream pipeline configuration field (e.g., the prompt template). We classify each failure mode as re-scoring or re-generation in App. F.2.

At the cross-benchmark level, we use PCA to characterize redundancy in task-level scores and Kendall's τ-b, together with pairwise rank inversions (App. H.3), to measure agreement among task-induced model rankings.

3 Experimental Setup

3.1 Benchmarks

Table 2 summarizes the eight benchmarks included in our audit. Together, they cover a broad range of cybersecurity evaluation tasks, including multiple-choice knowledge questions, vulnerability scoring, root-cause mapping, threat-actor attribution, attack-technique extraction, mitigation selection, and general cybersecurity reasoning. They draw on structured and semi-structured authoritative cybersecurity sources, including CVE, CWE, CVSS, MITRE ATT&CK, vulnerability advisories, and cyber threat intelligence reports (Sikos, 2023). We score each benchmark on its full released dataset

Table 2: Audited benchmarks with the number of scored tasks and questions.

Benchmark	Tasks	Questions	Reference
MMLU-CS	1	100	(Hendrycks et al., 2020)
CyberMetric	1	500	(Tihanyi et al., 2024)
SecBench	1	661	(Jing et al., 2024)
SecEval	1	2,189	(Li et al., 2023)
SECURE	3	2,502	(Bhusal et al., 2024)
CTI-Bench	5	4,610	(Alam et al., 2024)
AthenaBench	6	8,100	(Alam et al., 2025)
RedSage-Bench	5	30,000	(Suryanto et al., 2026)
Total	**23**	**48,662**	

unless the release structure or validation protocol requires a subset. For example, SecBench (Jing et al., 2024) reports 47,910 questions but releases 3,000, of which only 661 are English multiple-choice questions, so we score those 661. App. B.1 reports the published, released, and scored sizes for every benchmark.

3.2 Models

We evaluate 10 LLMs spanning proprietary, open-weight, and cybersecurity-specialized models, as summarized in Table 3. This mix allows us to assess whether reliability failures are specific to particular model classes or broader properties of the benchmark ecosystem. When a benchmark specifies an inference configuration, we preserve it as part of the original pipeline; otherwise, we use the fixed configuration implemented in our evaluation harness (§3.3). Open-weight models are served locally, while proprietary models are evaluated through their respective APIs.

Table 3: Evaluated LLMs by category; bracketed name segments are omitted in the text.

Model	Category	Reference
GPT-5.4	Proprietary	(Singh et al., 2025)
[Claude] Sonnet 4.6	Proprietary	(Anthropic, 2025)
Gemma-4[-31B]	Open-weight	(Google, 2026)
Qwen3.6[-35B]	Open-weight	(Yang et al., 2025)
Llama-3.3[-70B]	Open-weight	(Grattafiori et al., 2024)
GPT-OSS[-20B]	Open-weight	(Agarwal et al., 2025)
Primus-Nemotron[-70B]	Cybersecurity	(Yu et al., 2025)
Primus-Merged[-8B]	Cybersecurity	(Yu et al., 2025)
Foundation-Sec[-8B]	Cybersecurity	(Yang et al., 2026)
RedSage-Qwen3[-8B-DPO]	Cybersecurity	(Suryanto et al., 2026)

3.3 Evaluation Harness

We implement a common harness to isolate evaluation artifacts across cybersecurity benchmarks.[3] It standardizes the nine pipeline configuration fields of §2 wherever benchmark semantics permit, changing the evaluation procedure but not the task: questions, gold answers, and the intended capability are unchanged. The harness records the full evaluation trace (raw outputs, extracted predictions, question-level and aggregate scores, invalid-response rates, and pipeline configuration), making original-to-standardized differences explicit and reproducible.

App. D documents each standardization field by field, distinguishing corrections from supplied choices where specifications are incomplete. Table 12 reports defensible alternatives and their effects. Since output-format compliance is not an evaluation objective, we use a single pinned LLM judge for answer extraction and strict binary scoring with alias recognition (App. E.1).

4 Benchmark-Level Reliability Failures

We first audit reliability within individual benchmark pipelines, identifying 15 recurring failure modes spanning all five pipeline stages (Table 4). The main text highlights the highest-impact cases; Apps. G.1–G.5 provide additional evidence and per-model breakdowns.

4.1 Dataset Failures

$\mathcal{F}_1(\mathcal{D})$: Limited capability coverage. A domain benchmark should capture a meaningful range of the capabilities it is intended to assess rather than concentrating on a single form of competence. We therefore examine whether each benchmark covers both knowledge recall and analytical reasoning. Using a majority vote of four LLM classifiers, we classify each *question type* as either knowledge-oriented or analytical (App. G.1.1). Figure 1 shows that four benchmarks are highly skewed toward knowledge-oriented questions: RedSage-Bench, CyberMetric, MMLU-CS, and SecBench each contain

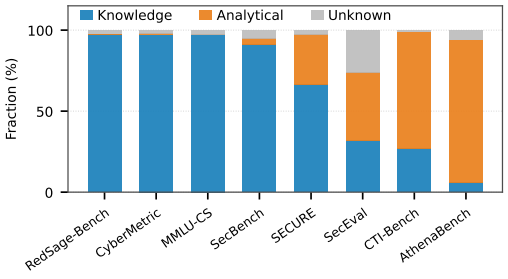


Figure 1: Knowledge vs. analytical question composition per benchmark (App. G.1.1).

[3] Audit code, with ablations and analyses, and the harness are publicly available. See §1 for links.

Table 4: Summary of the 15 recurring pipeline failure modes and their observed effects. *Affected units* reports where each failure was observed; maximum and median effects summarize the affected units when a per-unit distribution is available, and "—" indicates that a median is not meaningful. More information in App. F.6, App. G, and Table 16.

ID	Failure mode	Impact measure	Affected units	Maximum effect	Median effect
$\mathcal{F}_1(\mathcal{D})$	Limited capability coverage	Dominant question type	5 of 8 benchmarks	>95% of questions are of one type	
$\mathcal{F}_2(\mathcal{D})$	Gold-label correctness	Flag precision	7 of 8 benchmarks	23.8% of 998 checked flags	
$\mathcal{F}_1(\mathcal{P})$	Format-token leakage	Peer score gap	4 of 10 models	90.9 pp	81.3 pp
$\mathcal{F}_2(\mathcal{P})$	Prompt–question conflict	Single-letter response rate	8 of 10 models	33.0%	26.2%
$\mathcal{F}_3(\mathcal{P})$	Template incompatibility	Peer score gap	1 of 10 models	48 pp	—
$\mathcal{F}_1(\mathcal{I})$	Stop-sequence mismatch	Score gap	1 of 10 models	85.9 pp	—
$\mathcal{F}_2(\mathcal{I})$	Token-budget filter	Score gap	1 of 10 models	81 pp	—
$\mathcal{F}_3(\mathcal{I})$	Decoding drift	Score gap	1 of 10 models	40 pp	—
$\mathcal{F}_1(\mathcal{E})$	Extractor divergence	Extractor score gap	8 of 10 models	79.7 pp	1.3 pp
$\mathcal{F}_2(\mathcal{E})$	Denominator inflation	Score gap	3 benchmarks	99.8 pp	—
$\mathcal{F}_3(\mathcal{E})$	Metric-direction mismatch	Rank shift	10 of 10 models	5 ranks	1 rank
$\mathcal{F}_4(\mathcal{E})$	Prompt-mode sensitivity	Score spread	10 of 10 models	40 pp	7 pp
$\mathcal{F}_1(\mathcal{A})$	Logprob vs. generative scoring	Score gap	8 of 10 models	40.9 pp	2.8 pp
$\mathcal{F}_2(\mathcal{A})$	Task-level metric drift	Aggregation score gap	10 of 10 models	70 pp	33.5 pp
$\mathcal{F}_3(\mathcal{A})$	Aggregation inconsistency	Denominator gap	2 of 8 benchmarks	90.5 pp	—

more than 95% knowledge-oriented questions. AthenaBench shows the opposite pattern, with analytical questions comprising more than 95% of its questions. Their aggregate scores therefore capture only a narrow slice of domain capability and are dominated by performance on one question type rather than a balanced mixture of cybersecurity knowledge and reasoning.

$\mathcal{F}_2(\mathcal{D})$: Gold-label correctness. Noise in gold labels is non-negligible. Model-majority disagreement flags 1,140 questions, of which an automated search-grounded verifier checks 998 (App. G.1.2). It confirms 238 (23.8%) as label errors and 653 (65.4%) as false positives. Importantly, the 23.8% figure is the precision of the flagging procedure on this disagreement-selected set, not a benchmark-wide label-error rate. Model disagreement is therefore useful for triage, not for establishing correctness.

4.2 Prompt Failures

$\mathcal{F}_1(\mathcal{P})$: Format-token leakage. In CyberMetric, the answer-format template contains a literal placeholder that four of the 10 models misinterpret: Foundation-Sec and Primus-Nemotron produce mostly empty outputs, Gemma-4 reproduces the placeholder, and Primus-Merged continues the instruction text. Removing the placeholder restores valid responses for all four (App. G.2.1).

$\mathcal{F}_2(\mathcal{P})$: Prompt–question conflict. In SecEval, benchmark-level instructions to "select the correct answers" sometimes conflict with question wording that suggests a single-select response, leading models to return one choice for questions that require multiple selections. Under set-exact-match scoring, these responses receive zero credit even when the selected choice is partially correct. Across the eight models that produce parseable output, single-select responses occur on 10.1%–33.0% of the 927 multi-select questions (App. G.2.2).

$\mathcal{F}_3(\mathcal{P})$: Template incompatibility. In SECURE, Primus-Merged responds to MCQ questions with free-form explanations rather than the expected single- or multi-select answers. The benchmark extractor then treats the first answer-choice letter appearing in the explanation as the prediction, yielding a 48 percentage-point (pp) gap from peer models. This peer score gap reflects template incompatibility rather than task difficulty (App. G.2.3).

4.3 Inference Failures

$\mathcal{F}_1(\mathcal{I})$: Stop-sequence mismatch. The largest inference-stage effect happens in RedSage-Bench, where the official stop sequence, the newline character "\n", is triggered within Qwen3.6's reasoning preamble before any answer token is produced. Generating until the end-of-sequence token, allowing

sufficient output tokens for the reasoning span to close, and stripping that span before extraction increases the score by 85.9 pp (App. G.3.1).

$\mathcal{F}_2(\mathcal{I})$**: Token-budget filter.** In SecEval, the five-token output budget is below the 16-token minimum enforced by the Azure OpenAI endpoint we use for GPT-5.4, so every request fails before generating. The reported 0.3% comes from accidental letter matches inside the returned error payloads; raising the budget to 16 tokens yields 81.4% (App. G.3.2).

$\mathcal{F}_3(\mathcal{I})$**: Decoding drift.** In CyberMetric, the paper specifies sampling with temperature 1.0, top-p 0.9, and top-k 50, whereas the released script sets none inheriting backend defaults. For Primus-Merged, this discrepancy changes output-format compliance and results in 40 pp score gap, where 274 of the 500 questions change correctness between the two configurations (App. G.3.3). The effect is therefore driven largely by generation behavior and format compliance rather than knowledge.

4.4 Extraction and Scoring Failures

$\mathcal{F}_1(\mathcal{E})$**: Extractor divergence.** Extraction rules can disagree when outputs deviate from the expected format. In CTI-Bench vulnerability scoring, the prompt asks for the CVSS vector on the final line, while the released extractor takes the last vector found anywhere in the response. These rules disagree on 30.4% of model-question pairs where at least one extracts a vector, and on 96.5% of Primus-Merged's pairs (App. G.4.1). RedSage-Bench introduces a related ambiguity by reporting three metrics on the same generations without designating one as canonical. Primus-Merged scores 0.3% under exact match but 80.0% under prefix match, resulting in a 79.7 pp score gap (App. G.4.1).

$\mathcal{F}_2(\mathcal{E})$**: Denominator inflation.** In CTI-Bench's Root-Cause Mapping (CTI-RCM) task, invalid predictions, such as unparseable responses, are excluded from the denominator. Gemma-4 produces only two parseable outputs out of 1,000, which are both correct. Correct-over-valid scoring therefore reports 100.0%, whereas correct-over-total scoring reports 0.2%, a 99.8 pp difference and $500\times$ inflation (Example 10; App. G.4.2). AthenaBench's vulnerability-scoring task exhibits the same mechanism: unparseable CVSS vectors are excluded from the released denominator, affecting nine of the 10 models. Re-scoring the same generations while retaining these predictions and assigning the maximum deviation lowers scores by up to 85.4 pp, with a median decrease of 55.9 pp across the affected models (App. G.4.2).

$\mathcal{F}_3(\mathcal{E})$**: Metric-direction mismatch.** CTI-Bench and AthenaBench both evaluate CVSS-vector prediction, but CTI-Bench reports mean absolute deviation (MAD; lower is better), while AthenaBench transforms MAD into a normalized percentage (higher is better; App. G.4.3). The mismatch shifts model rankings by up to five positions across 10 models, so the two tasks are not directly comparable.

$\mathcal{F}_4(\mathcal{E})$**: Prompt-mode sensitivity.** In AthenaBench's Attack-Technique Extraction (Athena-ATE) task, the extractor reads only the final line of the response. A Chain-of-Thought (CoT) model may identify the correct techniques in its reasoning but omit them from the final line, causing correct evidence to be ignored. Holding the task and extractor fixed, we compare zero-shot, few-shot, and CoT prompting on the same samples. The resulting Athena-ATE scores differ by up to 40 pp across the 10 models (App. G.4.4). As each prompt mode generates new responses, we treat this as prompt-mode sensitivity rather than an extraction failure.

4.5 Aggregation Failures

$\mathcal{F}_1(\mathcal{A})$**: Logprob vs. generative scoring.** MCQ scores can change substantially depending on how answers are scored. Log-probability scoring ranks the answer choices directly without generating a response, while generative scoring extracts an answer from generated text. On RedSage-Bench, Gemma-4 scores 45.7% under generative regex extraction but 86.6% under log-probability scoring. Qwen3.6 shows the opposite pattern, scoring 85.9% and 59.2%, respectively (App. G.5.1). These differences are not caused by the RedSage-Bench stop sequence, since the generative scores use stop-free outputs. Neither scoring method is uniformly preferable because they measure different interactions between the model and evaluator.

$\mathcal{F}_2(\mathcal{A})$**: Task-level metric drift.** In attacker-attribution, scores vary substantially with the scoring rules. CTI-Bench provides strict exact-match scoring and a lenient variant that additionally credits alias-connected or related threat actors (e.g., APT28 and Fancy Bear). AthenaBench instead uses a strict binary verdict. Across the 10 models, differences among these task-level scores reach 70 pp (App. G.5.2). Aggregating task scores defined under different scoring rules can make benchmark-level scores and rankings depend on those rules rather than on a consistent measure of capability.

$\mathcal{F}_3(\mathcal{A})$**: Aggregation inconsistency.** Aggregation becomes inconsistent when scores computed over different populations are combined or compared as if they measured the same quantity. Gemma-4 illustrates the effect. CTI-RCM reports 100.0%, but only two of the model's 1,000 predictions are considered valid and both are correct (App. G.4.2). SecEval reports 9.5%, producing an apparent 90.5 pp gap. Its score is also affected by prompt wording that leads Gemma-4 to return single-select answers on multi-select questions, with a 33.0% single-select rate on this subset (App. G.2.2). Using a correct-over-total denominator reduces CTI-RCM to 0.2%. Under the standardized pipeline, the two scores become 70.9% and 78.3%, respectively (App. G.5.3). Aggregation can therefore propagate upstream pipeline inconsistencies into benchmark-level comparisons.

Implications for Model Comparison. Pipeline failures are widespread: every audited benchmark has at least two documented failure-mode interactions (Table 1). Large score changes can occur without changing the dataset or model, driven instead by configuration choices spanning all pipeline stages. But does this pipeline sensitivity also change comparative conclusions about models?

5 Cross-Benchmark Instability

We next ask whether this instability affects model comparison, analyzing the 23×10 task-by-model score matrix from three perspectives: score-level redundancy, rank-level disagreement, and ranking shifts under standardization.

5.1 Score-Level Redundancy

We first ask whether the 23 tasks provide independent evidence about model capability or largely measure the same underlying performance dimension. As shown in Figure 2, PCA of the column-standardized 23×10 accuracy matrix shows that the first component explains 95.25% of the variance. Thus, most tasks separate stronger from weaker models along one dominant performance axis rather than capturing distinct cybersecurity capabilities. This redundancy limits what broader task coverage can establish: adding tasks may increase benchmark scale without adding comparable measurement diversity, and aggregate scores may therefore provide limited evidence for fine-grained cybersecurity specialization. A Mantel test finds a moderate association between task-content similarity and task-induced rank agreement ($r = 0.516$; App. H.4), suggesting that semantic overlap explains only part of the observed ranking similarity. This is in line with the dominant performance axis reflecting more than repeated or similar question content.

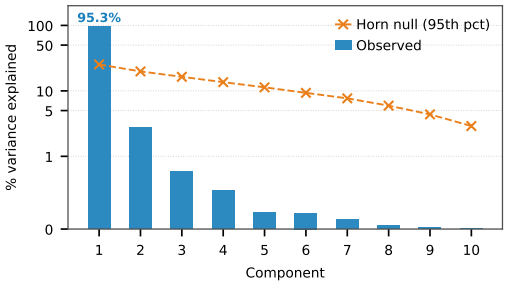


Figure 2: PCA scree plot of the task-by-model accuracy matrix with Horn's null (App. H.1).

5.2 Rank-Level Disagreement

Score-level redundancy does not imply stable model rankings. We measure rank agreement using Kendall's τ-b, which accounts for ties, together with pairwise rank inversions (Apps. H.2–H.3). Tasks may broadly agree on stronger and weaker models while disagreeing on specific model orderings. This pattern is clearest for semantically similar tasks. CTI-Bench and AthenaBench both evaluate vulnerability scoring and attacker attribution, yet they rank the same 10 models differently. The rank agreement is weak for vulnerability scoring (τ-b=0.29) and attacker attribution (τ-b=0.24). The vulnerability-scoring pair also reorders models by up to five positions (App. H.3). These differences

Table 5: Rank shifts $\Delta r_b(m)$ under standardization (positive: upward). Bottom row: Spearman's ρ between original and standardized rankings; 95% bootstrap confidence intervals in App. I.4.

Model	MMLU-CS	SecEval	SECURE	CTI-Bench	AthenaBench	CyberMetric	RedSage-Bench	SecBench
GPT-5.4	+1	0	0	0	0	0	0	-1
Sonnet 4.6	+5	-5	0	0	0	0	0	0
Gemma-4	+6	+7	+4	+3	+7	+5	+5	+6
Qwen3.6	+6	+3	-1	-2	-3	0	+1	+4
Llama-3.3	-3	-5	-3	-2	-2	-2	0	-3
GPT-OSS	+5	+4	0	-5	-5	-1	+2	+1
Primus-Nemotron	-7	-1	+4	+3	+2	+2	-2	-1
Primus-Merged	-3	-1	0	0	0	-1	-3	-1
Foundation-Sec	-6	-5	-4	0	+2	0	-1	-4
RedSage-Qwen3	-4	+3	0	+3	-1	-3	-2	-1
Spearman ρ	-0.47	0.03	0.65	0.64	0.42	0.73	0.71	0.50

coincide with incompatible extraction rules, alias handling, and scoring conventions. So, comparative claims, such as "m_i outperforms m_j at vulnerability scoring," can depend on the pipeline.

5.3 Ranking Shifts Under Standardization

Using the evaluation harness (§3.3), we standardize the nine pipeline configuration fields wherever benchmark semantics allow. App. I.1 specifies which fields are standardized for each benchmark, and App. I.2 reports the underlying scores.

For benchmark b and model m, let $\Delta r_b(m) = r_b^{\text{orig}}(m) - r_b^{\text{std}}(m)$ denote the change in rank between the original and standardized pipelines, with rank 1 the highest-scoring model and ties broken by a stable sort (App. I.3). Positive values indicate upward movement after standardization. Table 5 shows that nine of the 10 models shift by at least three ranks on at least one benchmark; only GPT-5.4 does not. Gemma-4 rises on every benchmark where its rank changes, consistent with the extraction and denominator effects in §4. Because standardization changes upstream fields like the prompt formatting, decoding, and token budget, these scores require re-generation, not just re-scoring.

The original and standardized rankings also show substantial disagreement (Table 5). Spearman's ρ is -0.47 for MMLU-CS and 0.03 for SecEval, indicating strong reordering, and ranges from 0.42 to 0.73 on the other benchmarks. On MMLU-CS, all four cybersecurity-specialized models shift downward by three to seven positions, while Gemma-4 shifts upward by six.

These shifts are robust to question-sampling variability: across 5,000 paired bootstrap replicates, every rank shift of at least 3 positions keeps its direction in at least 97.7% of samples, and every 95% interval for Spearman's ρ excludes 1, so the reordering persists under question resampling (App. I.4).

Implications for Benchmark Design. Broader task coverage does not guarantee more informative or stable model comparisons. Many tasks provide redundant score-level evidence, while model rankings remain sensitive to pipeline choices. Reliable benchmarking therefore requires both diverse capability coverage and explicit, consistent evaluation pipelines.

6 Toward Reliable Benchmarking

A benchmark release should specify the full measurement pipeline that produced its scores, not only the dataset and metric: label provenance, the exact prompt and chat template, inference settings, the executable extractor or judge, scoring and denominator rules, aggregation, and reliability statistics such as invalid-response rates (App. J.2). Harnesses should pin model, serving, and judge configurations; record model IDs and run timestamps for closed models; and store raw outputs so downstream changes can be re-scored without requiring new generations.

Some failures cannot be fixed automatically because benchmark intent is underspecified, such as whether partial credit applies or which aliases count as correct. Harnesses should make such decisions explicit and executable. Benchmarks that evaluate similar tasks should align metric direction, scoring rules, alias handling, and aggregation, or else report their scores as distinct measurements. In evolving

domains such as cybersecurity, gold labels should be periodically audited against authoritative sources. App. J.1 discusses which parts of the audit can be automated and which require manual judgment.

7 Related Work

Large-scale frameworks such as BIG-bench (Srivastava et al., 2022), HELM (Liang et al., 2023), and DynaBench (Kiela et al., 2021) have made benchmark-based comparison central to LLM evaluation. Recent work shows that reported performance can be sensitive to prompt wording, decoding parameters, evaluator design, and extraction rules (Shi et al., 2024; Sun et al., 2024; Wang et al., 2023). We build on this work by modeling benchmarks as end-to-end measurement pipelines and auditing failures across all five stages. Prior work also shows that aggregate score correlations can obscure disagreement in model rankings (Perlitz et al., 2024). We extend this perspective to cybersecurity, where even semantically similar tasks can rank the same models differently when their evaluation conventions differ.

Bean et al. (2025) study the complementary problem of *construct validity*: whether benchmarks measure the phenomena they claim to measure and support the resulting claims. Their review spans multiple benchmark domains but does not include cybersecurity benchmarks. Böhme (2026) questions whether strong benchmark results are sufficient evidence of technological progress, pointing to overfitting and unreliable measures of effectiveness, and calls for evaluations that explicitly identify the limitations of new techniques. We instead focus on the *measurement reliability* of executable evaluation pipelines and provide a systematic audit of this problem in cybersecurity. A benchmark can be reproducible and internally consistent while still failing to measure the real-world capability it claims to represent, so reliability is necessary but not sufficient for valid evaluation.

8 Limitations

Our empirical findings are specific to cybersecurity. We chose this domain because it makes pipeline failures unusually observable. Much of its ground truth can be checked against authoritative sources such as CVE, CWE, CVSS, and MITRE ATT&CK (App. G.1.2). Many answers are also structured, so certain failures such as an invalid CVSS vector can be detected mechanically. Finally, the benchmark ecosystem is heterogeneous: 44 of 72 pipeline configuration fields cannot be reproduced from documentation alone (App. C). Mechanical failures involving token budgets, stop sequences, extraction, denominators, and aggregation may transfer to other domains, which we do not test directly. Other benchmarks, models, backends, and evaluators may show failures outside our taxonomy. Some benchmark artifacts are also underspecified, requiring us to make implementation choices. We document these choices in the paper and appendix, but alternative interpretations may be defensible.

Some comparative conclusions also depend on the four judgment-dependent standardization choices in Table 12; each admits a defensible alternative, so those conclusions are conditional on the convention adopted. Our standardized extraction also uses a pinned LLM judge. An independent judge agrees on 99.6% of verdicts and produces nearly identical rankings (App. E.3), reducing but not eliminating concerns about judge dependence. We quantify question-sampling variability with a question-level bootstrap (App. I.4), but do not estimate variability across repeated model generations. Gold-label and judge decision verification are also LLM-assisted, with manual validation on random stratified samples of questions drawn across each benchmark's tasks (Apps. E.2 and G.1.2).

9 Conclusion

We audited cybersecurity LLM benchmarks as end-to-end measurement pipelines. Across eight benchmarks, 23 tasks, and 10 LLMs, we identified 15 recurring failure modes spanning all five pipeline stages. Individual pipeline choices can shift scores by more than 80 pp, nine of the 10 models shift by at least three ranks on at least one benchmark under standardization, and semantically similar tasks can rank the same models differently. Thus, benchmark scores should be treated as pipeline-dependent measurements, supported by meaningful capability coverage and explicit, consistent evaluation pipelines.

Ethics Statement

This work studies the reliability of cybersecurity benchmarks rather than the development of new offensive capabilities. The audited tasks cover vulnerabilities, attack techniques, malware behavior, and threat intelligence, but are drawn from existing public benchmarks and authoritative cybersecurity sources. We do not introduce new exploit procedures, malware implementations, offensive datasets, or attack automation.

The main ethical concern is the harm caused by unreliable evaluation. In a high-stakes domain such as cybersecurity, unstable benchmark pipelines can support misleading claims about model capability, safety, or specialization. Such claims may influence model selection, deployment decisions, and trust in automated security systems. By exposing pipeline failures and making evaluation choices explicit, our goal is to improve transparency, reproducibility, and scientific reliability.

A secondary concern is dual use. Detailed descriptions of benchmark failure modes could facilitate benchmark-specific optimization or gaming. We therefore focus on evaluation mechanisms rather than methods for exploiting security systems, and our public reporting emphasizes aggregate results and pipeline metadata rather than unnecessary question-level content.

Reproducibility Statement

App. B specifies the evaluation scope, including the reported, released, and scored size of every benchmark (App. B.1), the task inventory (App. B.2), the serving stack, decoding configuration, and evaluation dates (App. B.3), the prompt templates (App. B.4), and the per-run configuration record stored by the harness (App. B.5). The pipeline ledger (App. D.4) documents every benchmark-specific pipeline resolution, including the released behavior, the setting we use, and the measured effect of each change, and Table 10 lists the pipeline configuration fields that cannot be reproduced from benchmark documentation alone. The pinned judge model, prompt, decoding settings, and failure handling are given in App. E.1, together with the judge validation and independent-judge results (Apps. E.2–E.3). The audit protocol and the distinction between re-scoring and re-generation analyses are described in App. F, the gold-label verification procedure in App. G.1.2, and the bootstrap procedure for rank stability in App. I.4. The audit code, the evaluation harness, raw model generations, run logs, and analysis notebooks are released with the paper (§1; Table 6).

The harness is released on PyPI as an open-source Python package for model-agnostic evaluation of cybersecurity LLMs.[4] It provides a common interface for hosted APIs and locally served models. Open-ended responses can be scored using a configurable LLM judge, allowing the evaluated model and judge model to be selected independently. For transparency, an adapter exports aggregate scores and pipeline metadata to EveryEvalEver (Batzner et al., 2026), while withholding question-level content in accordance with our Ethics Statement.

AI Use Statement

LLMs are the object of study in this work and are also used as instruments within the audit methodology, as described in the main text and appendix. A pinned LLM judge (GPT-5.4) performs answer extraction and binary grading in the standardized pipeline (App. E.1), and an independent judge (Sonnet 4.6) re-grades stored outputs to test judge dependence (App. E.3); both are validated against human annotation (App. E.2). Four LLM classifiers label benchmark questions as knowledge-oriented or analytical (App. G.1.1), and a search-grounded LLM verifier triages candidate gold-label errors, whose decisions are validated by two human annotators (App. G.1.2). All quantitative claims in the paper are computed by the released harness and audit code from stored model outputs; no LLM was used to formulate, prove, or interpret the paper's claims beyond these documented roles. LLMs were also used to assist with editing and polishing the manuscript text; all content was reviewed and verified by the authors.

[4] *link withheld for double-blind review*

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. GPT-OSS-120B & GPT-OSS-20B Model Card. *arXiv preprint arXiv:2508.10925*, 2025.

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.

Md Tanvirul Alam, Dipkamal Bhusal, Salman Ahmad, Nidhi Rastogi, and Peter Worth. AthenaBench: A Dynamic Benchmark for Evaluating LLMs in Cyber Threat Intelligence. *arXiv preprint arXiv:2511.01144*, 2025.

Anthropic. System Card: Claude Sonnet 4.6, 2025. URL https://www.anthropic.com/news/claude-sonnet-4-6.

Jan Batzner, Sree Harsha Nelaturu, Damian Stachura, Anastassia Kornilova, Jon Crall, Tommaso Cerruti, Yanan Long, Yifan Mai, Sanchit Ahuja, Asaf Yehudai, et al. Every eval ever: A unifying schema and community repository for AI evaluation results. *arXiv preprint arXiv:2606.14516*, 2026.

Andrew M Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, Jan Batzner, Negar Foroutan Eghlidi, Chris Schmitz, Karolina Korgul, Hunar Batra, et al. Measuring what matters: Construct validity in large language model benchmarks. *Advances in Neural Information Processing Systems 38*, pp. 19868–19949, 2025.

Dipkamal Bhusal, Md Tanvirul Alam, Le Nguyen, Ashim Mahara, Zachary Lightcap, Rodney Frazier, Romy Fieblinger, Grace Long Torales, Benjamin A Blakely, and Nidhi Rastogi. SECURE: Benchmarking large language models for cybersecurity. In *2024 Annual Computer Security Applications Conference (ACSAC)*, pp. 15–30. IEEE, 2024.

Marcel Böhme. Keynote talk: Adversarial benchmarking: An urgent call for a stronger signal of technological progress. In *IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2026.

Avijit Ghosh, Anka Reuel, Jenny Chim, Wm Matthew Kennedy, Srishti Yadav, Jennifer Mickel, Yanan Long, Andrew Tran, Anastassia Kornilova, Damian Stachura, et al. Evaluation cards: An interpretive layer for AI evaluation reporting. *arXiv preprint arXiv:2606.09809*, 2026.

Google. Gemma 4 Model Card, 2026. URL https://ai.google.dev/gemma/docs/core/model_card_4.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Pengfei Jing, Mengyun Tang, Xiaorong Shi, Xing Zheng, Sen Nie, Shi Wu, Yong Yang, and Xiapu Luo. SecBench: A comprehensive multi-dimensional benchmarking dataset for LLMs in cybersecurity. *arXiv preprint arXiv:2412.20787*, 2024.

Douwe Kiela, Max Bartolo, Yixin Nie, et al. Dynabench: Rethinking Benchmarking in NLP. In *Proceedings of NAACL*, 2021.

Guancheng Li, Yifeng Li, Wang Guannan, Haoyu Yang, and Yang Yu. SecEval: A Comprehensive Benchmark for Evaluating Cybersecurity Knowledge of Foundation Models. *GitHub*, 2023.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=iO4LZibEqW.

Yotam Perlitz, Ariel Gera, Ofir Arviv, Asaf Yehudai, Elron Bandel, Eyal Shnarch, Michal Shmueli-Scheuer, and Leshem Choshen. Benchmark Agreement Testing Done Right: A Guide for LLM Benchmark Evaluation. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2024.

Cheng Shi, Zhisheng Zhang, Yujiu Yang, et al. A Thorough Examination of Decoding Methods in the Era of Large Language Models. In *Proceedings of EMNLP*, 2024.

Leslie F. Sikos. Cybersecurity Knowledge Graphs. *Knowledge and Information Systems*, 65(9): 3511–3531, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. OpenAI GPT-5 System Card. *arXiv preprint arXiv:2601.03267*, 2025.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. *Transactions on Machine Learning Research*, 2022.

Jiuding Sun, Chantal Shaib, and Byron Wallace. Evaluating the Zero-Shot Robustness of Instruction-Tuned Language Models. In *International Conference on Learning Representations*, volume 2024, pp. 48103–48141, 2024.

Naufal Suryanto, Muzammal Naseer, Pengfei Li, Syed Talal Wasim, Jinhui Yi, Juergen Gall, Paolo Ceravolo, and Ernesto Damiani. RedSage: A Cybersecurity Generalist LLM. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=W4FAenIrQ2.

Norbert Tihanyi, Mohamed Amine Ferrag, Ridhi Jain, Tamas Bisztray, and Merouane Debbah. CyberMetric: A Benchmark Dataset Based on Retrieval-Augmented Generation for Evaluating LLMs in Cybersecurity Knowledge. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 296–302. IEEE, 2024.

Peiyi Wang, Lei Li, Liang Chen, et al. Large Language Models are not Fair Evaluators. *arXiv preprint arXiv:2305.17926*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhuoran Yang, Ed Li, Jianliang He, Aman Priyanshu, Baturay Saglam, Paul Kassianik, Sajana Weerawardhena, Anu Vellore, Blaine Nelson, Neusha Javidnia, et al. Llama-3.1-FoundationAI-SecurityLLM-Reasoning-8B Technical Report. *arXiv preprint arXiv:2601.21051*, 2026.

Yao-Ching Yu, Tsun-Han Chiang, Cheng-Wei Tsai, Chien-Ming Huang, and Wen-Kwang Tsao. Primus: A Pioneering Collection of Open-Source Datasets for Cybersecurity LLM Training. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10402–10424, 2025.

A Where to Find Each Artifact

Table 6 provides a roadmap to the artifacts supporting our pipeline audit and standardization decisions. Materials that are impractical to include in the paper, such as raw generations, run logs, plotting code, and interactive notebooks, are released in the audit repository, which is also linked in the table.

Four box types recur throughout the appendix. A *prompt* box reproduces model or judge instructions. An *example* box shows a concrete failure-mode instance, including the prompt, raw output, evaluator behavior, and consequence. A *case* box presents a verified gold-label case grounded in authoritative sources. A *code* box shows an executable harness artifact. Prompt, example, case, and code boxes are numbered by type.

Table 6: Roadmap to the artifacts supporting the benchmark audit. The appendix contains the information needed to assess the reported pipeline decisions; larger execution artifacts are provided in the audit repository.

Artifact	Location
Pipeline specification	
Original and standardized pipelines, by benchmark	App. D.4; Table 11
Specification status of all 72 pipeline fields	App. C; Table 10
Standardizations with defensible alternatives	App. D.2, Table 12
Field-level pipeline ledger and measured effects	App. D.4, Table 13
Harness configuration	App. D
Audit protocol	
Audit procedure	§2.1; App. F.1
Stage-by-benchmark audit coverage	App. F.5; Table 18
Re-scoring, re-generation, and automation limits	App. F.2; Table 16
Evidence and results	
Evidence for each failure mode	App. G
Original and standardized scores	App. I.2; Table 30
Gold-label audit and verified cases	Apps. G.1.2–G.1.3
Benchmark sizes and task inventory	Tables 7, 8
Repository-only artifacts	
Raw generations, run logs, plots code, and notebooks	Audit repository

B Evaluation Setup

This section provides the implementation details needed to reproduce our evaluation. We first define the scored benchmark subsets and task inventory. We then describe the serving environment, prompt construction, and the configuration records produced by the evaluation harness. Benchmark-specific pipeline choices and their standardization are documented separately in App. D.

B.1 Evaluation Scale and Sampling

Table 7 distinguishes three dataset sizes for each benchmark. *Reported* is the number of questions stated by the benchmark paper or release documentation for the corresponding evaluation scope. *Released* is the number of questions available in the public artifact. *Scored* is the subset evaluated in our audit. The scored subsets sum to 48,662 questions across 23 tasks, which is the evaluation scope reported in the main presentation. Per-task counts are given in Table 8.

Table 7: Benchmark sizes used to define the audit scope. Per-task scored sizes are given in Table 8.

	MMLU-CS	SecEval	SECURE	CTI-Bench	Athena-Bench	Cyber-Metric	RedSage-Bench	SecBench	Total
Reported	100	2,126	4,068	4,947	8,100	10,000	30,240	47,910	
Released	100	2,189	4,068	5,610	8,100	10,180	30,290	3,000	
Scored	100	2,189	2,502	4,610	8,100	500	30,000	661	**48,662**

Only three benchmarks have matching reported and released sizes. Several of the remaining differences arise from counting or release conventions rather than from our sampling. SecEval reports an overall total of 2,126 in its README, although its per-topic counts match the 2,189 distinct questions in the released file. CyberMetric reports 10,000 questions, while its release contains 10,180. CTI-Bench does not report a benchmark-wide total. Summing the paper's per-task figures gives 4,947, partly because CTI-ATE is described using 397 unique ATT&CK techniques rather than its 60 evaluation questions. For RedSage-Bench, the paper reports 30,240 questions, of which 240 are open ended Q&A. The released artifact also contains a separate 50-question validation split.

SecBench has the largest substantive difference between the reported and released data. The paper reports 47,910 questions, but only 3,000 are publicly released. These comprise 2,730 MCQs and 270 SAQs. Only 661 of the released MCQs are in English, and none of the SAQs are. We therefore evaluate the 661 English MCQs.

The scored set is also smaller than the release when a benchmark contains tasks or splits outside our evaluation scope. For SECURE, we evaluate three of its six tasks: MAET, CWET, and KCV. For CTI-Bench, we evaluate five tasks and exclude the separate 1,000 question CTI-RCM-2021 split. For RedSage-Bench, we use the 30,000 closed-form question test split and exclude the 240 open-ended questions and the 50-question validation split. For CyberMetric, we use the separately released 500-question set that its authors describe as human-validated. Finally, SECURE-CWET contains one all-empty row with no prompt; the harness drops this row before evaluation.

B.2 Task Inventory

The audit covers 23 tasks across the eight benchmarks. Table 8 gives the task identifier (ID) used in the harness, task-level metric, denominator policy, and number of scored questions for each task. These counts sum to the 48,662 questions, as reported in Table 7.

Table 8: Inventory of the 23 scored tasks. Metric names describe the task-level quantities recorded by the harness. The standardized denominator retains attempted questions rather than dropping unparseable model outputs. For TAA, both binary and partial-credit variants are retained for the audit, while standardized model comparisons use strict binary scoring (App. G.5.2). Benchmark-level summaries include only comparable, bounded, higher-is-better metrics; raw MAD, set-F1, and partial-credit attribution are excluded where appropriate (App. D.4).

Benchmark	Task ID	Metric	Denominator policy	Questions
MMLU-CS	`mmlu_cs`	Accuracy	Correct / total	100
SecEval	`seceval`	Set-exact-match accuracy	Correct / total	2,189
SECURE	`secure_maet`	Accuracy	Correct / total	1,072
	`secure_cwet`	Accuracy	Correct / total	964
	`secure_kcv`	Accuracy	Correct / total	466
CTI-Bench	`cti_mcq`	Accuracy	Correct / total	2,500
	`cti_rcm`	CWE accuracy	Correct / total	1,000
	`cti_vsp`	CVSS MAD	All questions	1,000
	`cti_ate`	Accuracy	Correct / total	60
	`cti_taa`	Binary / partial	All questions	50
AthenaBench	`ckt`	Accuracy	Correct / total	3,000
	`rms`	Question-level set-F1	All questions	500
	`athena_taa`	Binary / partial	All questions	100
	`athena_ate`	Accuracy	Correct / total	500
	`athena_rcm`	CWE accuracy	Correct / total	2,000
	`athena_vsp`	Normalized CVSS score	All questions	2,000
CyberMetric	`cybermetric`	Accuracy	Correct / total	500
RedSage-Bench	`frameworks`	Accuracy	Correct / total	5,000
	`generals`	Accuracy	Correct / total	5,000
	`skills`	Accuracy	Correct / total	10,000
	`cli`	Accuracy	Correct / total	5,000
	`kali`	Accuracy	Correct / total	5,000
SecBench	`secbench`	Set-exact-match accuracy	Correct / total	661
Total				**48,662**

Several task abbreviations recur throughout the appendix. Root-Cause Mapping (RCM) maps a CVE description to a CWE ID. Vulnerability Severity Prediction (VSP) predicts a CVSS vector. Attack-Technique Extraction (ATE) extracts MITRE ATT&CK technique IDs, while Threat-Actor Attribution (TAA) identifies the actor behind a threat-intelligence description. Response and Mitigation Selection (RMS) maps scenarios to ATT&CK mitigation IDs. SECURE's MAET and CWET tasks are multiple-choice tasks based on MITRE ATT&CK and CWE, respectively, while KCV is a true-or-false task over CVE records. AthenaBench's CKT is a five-option cybersecurity knowledge test.

The denominator column in Table 8 describes the standardized scoring rule. Empty or unparseable model answers remain in the evaluation population rather than being discarded. For accuracy tasks, these answers therefore count as incorrect. Task-specific handling for structured metrics such as VSP is described in App. G.4.2. For attacker attribution, the harness retains the alternative scoring variants needed for the audit, while standardized model comparisons use the strict binary convention described in App. G.5.2.

B.3 LLM Serving Stack

We serve all eight open-weight models locally with vLLM in `bfloat16` and without quantization. Each evaluation job uses four NVIDIA H200 141 GB GPUs, with tensor parallelism set to the number of visible devices. We set `gpu_memory_utilization` to 0.90 by default and to 0.85 for Qwen3.6 and RedSage-Qwen3, which require additional memory headroom. Table 9 reports the maximum context length used for each checkpoint.

Table 9: vLLM serving configuration for open-weight models. *Context* is the configured `max_model_len` in tokens. *Memory* is `gpu_memory_utilization`.

Checkpoint	Context	Mem. (%)
Qwen3.6-35B-A3B	32,768	85
RedSage-Qwen3-8B-DPO	16,384	85
Llama-3.3-70B-Instruct	8,192	90
Llama-Primus-Nemotron-70B	8,192	90
Foundation-Sec-8B-Instruct	8,192	90
gpt-oss-20b	8,192	90
gemma-4-31B-it	4,096	90
Llama-Primus-Merged	Default	90

The standardized inference configuration uses greedy decoding with temperature 0 and top-p 1.0. Local vLLM runs additionally use `min_tokens=50` to avoid empty end-of-sequence completions. No backend stop sequence is used in the standardized configuration. Maximum output length is calibrated by task, with 1,024 tokens as the default. These settings describe the standardized configuration. When reproducing an original benchmark pipeline or conducting a controlled failure-mode analysis, we instead use the benchmark-specific setting being studied, as documented in App. D.4.

Two analyses use sampled decoding. For SECURE, we additionally evaluate the documented temperature of 0.7 and pin seed 42. For the CyberMetric decoding-drift analysis, $\mathcal{F}_3(\mathcal{I})$, we compare the greedy control with the documented temperature of 1.0 and top-p of 0.9 (App. G.3.3). These sampled runs pin a seed; the main greedy runs do not depend on one.

GPT-5.4 and Sonnet 4.6 are evaluated through Azure-hosted endpoints (i.e., REST APIs). GPT-5.4 uses Azure OpenAI chat completion API with `api-version=2024-12-01-preview`. Sonnet 4.6 uses an Azure Anthropic-messages passthrough with `anthropic-version=2023-06-01`. Both are evaluated at temperature 0. Provider-side content filtering, such as guardrails, is disabled through the deployment configuration for both models. This avoids failed requests caused solely by the cybersecurity content of benchmark questions and makes pipeline behavior more reproducible.

The local runtime uses CUDA 12.1, Python 3.10, PyTorch, `transformers`, and vLLM. Open-weight checkpoints were retrieved at their then-current Hugging Face revisions, or from a fixed local checkpoint where applicable. We record the resolved checkpoint commit hashes with the released artifacts so that the evaluated weights can be recovered. Local inference was run from April 19–20, 2026, with two additional task runs from May 4–5, 2026. Hosted inference on Azure was run on May 5, and judge calls were performed between April 21 and May 5, 2026.

B.4 Prompt Templates

Prompt construction depends on both the benchmark and the task. The standardized harness preserves the benchmark question and intended answer semantics while making prompt and chat formatting explicit. Single-select multiple-choice tasks use the template shown in Prompt 1. Multi-select and

structured-output tasks use task-specific output instructions instead. Benchmark-specific system prompts are retained where they define the task presentation. Representative system prompts for CTI-Bench and SecEval are shown in Prompt 2 and Prompt 3, respectively. MMLU-CS and SecEval also prepend fixed few-shot exemplar blocks when required by the corresponding pipeline. The complete benchmark-by-benchmark prompt provenance is given in App. D.4.

PROMPT 1 : Shared single-select MCQ user template

```
You are given multiple choice questions. Answer with the option letter (A, B, C, D) from the given choices
directly.

Question: {question}
{choices}
Answer:
```

PROMPT 2 : CTI-Bench system prompt

```
You are a cybersecurity expert specializing in cyberthreat intelligence.
```

PROMPT 3 : SecEval system prompt

```
Below are multiple-choice questions concerning cybersecurity. Please select the correct answers and respond
 with the letters ABCD only.
```

B.5 Reproducible Configuration

Each run stores the configuration needed to interpret its reported score. This includes the relevant pipeline configuration fields defined in App. D, together with model, judge, backend, task, and schema provenance. In particular, the record preserves the decoding configuration, token budget, stop-sequence handling, extraction and scoring rules, denominator policy, and prompt configuration associated with the run. This allows each reported score to be traced back to the pipeline that produced it. A representative configuration and policy record is shown in Code 1. The complete benchmark-specific values and their provenance are given in App. D.4.

CODE 1 : Harness config and per-run policy record

```
# decoding / token budget
temperature = 0.0
top_p       = 1.0
max_tokens  = per-task-calibrated
min_tokens  = 50    # local vLLM only
answer_stop = None

# scoring policies
denominator_policy =
  "accuracy = correct / total over all attempted questions;
   unparseable/empty model answers count as incorrect;
   judge-API failures are excluded from both numerator and denominator."

think_handling =
  "strip <think>...</think> before judging;
   if an answer-level stop is configured, apply it after removing reasoning."

scoring =
  "llm-as-judge: one call performs answer extraction and returns a CORRECT/INCORRECT verdict."

# provenance stored with each result:
# model {name, provider, endpoint}
# judge {name, provider}
# tasks []
# backend
# run_timestamp
# schema_version
```

B.6 SOFTWARE

The released harness, cyeval,[5] is distributed via PyPI and can run inference and judging through a common command-line interface. Hosted models and locally served OpenAI-compatible endpoints use the same evaluation workflow. Code 2 shows the basic workflow.[6] These commands illustrate package usage and are not intended to reproduce the exact experimental configuration used in this paper. Exact run configurations and provenance are provided with the released audit repository.[7]

CODE 2: Installing and running the harness

```
pip install cyeval

# inference + judge
cyeval run \
  --tasks mcq seceval vsp taa \
  --model  openai/gpt-4o \
  --judge  anthropic/claude-sonnet-4-6 \
  --output-dir outputs/run1

# local vLLM endpoint as the model under test
vllm serve Qwen/Qwen3-8B --port 8000 --enforce-eager
cyeval run --tasks mcq \
  --model hosted_vllm/Qwen/Qwen3-8B \
  --base-url http://localhost:8000/v1 \
  --api-key EMPTY \
  --output-dir outputs/qwen
```

C PIPELINE SPECIFICATION GAPS

Across eight benchmarks and nine pipeline configuration fields, we inspect 72 field–benchmark pairs. We classify each pair by comparing the benchmark documentation with its released implementation. A field is *undefined* when the documentation does not specify it, *contradicted* when the documented and released layers do not define the same executable behavior, and *matched* when they agree. Of the 72 pairs, 36 are undefined, 8 are contradicted, and only 28 are specified and matched. Thus, 44 of 72 pipeline configuration fields (61%) cannot be reproduced from benchmark documentation alone. Table 10 gives the complete field-level breakdown.

Table 10: Specification status of the nine pipeline configuration fields across the eight benchmarks. A green check indicates that a field is specified in the documentation and *matched* by the released implementation. A gray dot indicates an *undefined* field, and a red cross indicates that the documented and released layers do not define the same executable behavior, so one has *contradicted* the other. *Gaps* counts undefined and contradicted fields. Overall, 44 of 72 fields (61%) are gaps; stop sequences are undefined for all eight benchmarks.

	Prompt		Inference			Extraction			Aggregation	
Benchmark	Template	Chat	Decoding	Max tokens	Stop	Extractor	Scoring	Denominator	Rule	**Gaps**
MMLU-CS	✓	✓	✓	✓	·	✓	✓	✓	✓	1
SecEval	·	·	·	·	·	✓	✓	·	✓	6
SECURE	✓	·	×	·	·	✓	✓	·	·	6
CTI-Bench	✓	·	✓	·	·	✓	×	·	✓	5
AthenaBench	✓	·	✓	·	·	✓	×	·	✓	5
CyberMetric	×	✓	×	·	·	✓	✓	·	✓	5
RedSage-Bench	·	×	×	×	·	·	✓	·	·	8
SecBench	·	·	·	·	·	·	✓	·	·	8

This underspecification matters because an unspecified field must be supplied by the evaluator. Different choices can produce different prompts, generations, extracted answers, scores, or aggregations even when the benchmark questions and evaluated model are unchanged. The same ambiguity also

[5] https://anonymous.4open.science/r/cyeval-DF7B/

[6] This install command does not work during review because the package name is anonymized. Install the review copy from its downloaded folder with `pip install -e .`; the other commands are unchanged.

[7] https://anonymous.4open.science/r/audit-F356/

affects our reconstruction: when neither the documentation nor released implementation uniquely determines a field, our selected value is an explicit evaluation choice rather than a uniquely correct setting. We record these choices in the *pipeline ledger*, defined in Table 13, instead of treating them as part of benchmark specification.

The eight contradicted fields illustrate the different ways in which the documented and released layers can diverge:

- **CTI-Bench scoring rule:** the documentation describes exact-match scoring, while the released TAA scorer also credits alias-connected and related threat actors.
- **AthenaBench scoring rule:** the VSP normalization constant R=7.7 appears in `config.yaml` but is absent in the paper, so the reported metric cannot be reconstructed from the paper alone.
- **SECURE decoding:** the paper specifies temperature T=0.7, but no inference implementation is released to enact that setting.
- **CyberMetric prompt template:** the prompt in the README differs from the prompt hard-coded in the released evaluator.
- **CyberMetric decoding:** the paper specifies T=1.0, top-p 0.9, and top-k 50, while the released evaluator sets none of these parameters and therefore inherits backend defaults.
- **RedSage-Bench chat formatting:** the two released inference examples use incompatible chat-formatting configurations.
- **RedSage-Bench decoding:** the two released inference examples specify different temperatures.
- **RedSage-Bench maximum output tokens:** the two released inference examples specify different token budgets.

The pipeline ledger also reports measured effects for fields that we perturb. These are *isolated single-field counterfactuals*: we vary one pipeline configuration field on the same questions while holding the remaining fields fixed. Downstream changes, such as extraction or denominator policy, can be evaluated by re-scoring stored outputs. Upstream changes, such as prompts or inference settings, require re-generation (App. F.2). These effects are therefore distinct from the original-to-standardized differences in Table 30, which reflect the combined change from the original pipeline to the standardized pipeline.

For each original pipeline, the ledger records the extractor used to reproduce its released behavior. When no extractor is released, we explicitly document the one we supply. The standardized pipeline instead uses the pinned LLM-based extraction and scoring rules described in App. E.1. Some failures cannot be isolated with a controlled counterfactual because the original pipeline cannot be executed without the failure or because the effect depends on an interaction between model output style and the evaluator. In these cases, we report a clearly labeled *peer-gap estimate*, defined as the median score of unaffected models minus the affected model's score. A ledger entry marked *not isolated* indicates that the field was supplied or changed but was not independently ablated.

D Benchmark Pipelines

This section documents how each benchmark pipeline is reconstructed and standardized. We distinguish three sources of information. The *documented* layer is the behavior stated in the benchmark paper or documentation. The *released* layer is the behavior implemented by the public artifact. When these layers are incomplete or inconsistent, we explicitly record the resolution used in our audit. We then identify the settings used by the standardized harness. This separation is important because a supplied or normalized setting is an evaluation choice, not necessarily a uniquely correct interpretation of the benchmark.

D.1 Pipeline Summary

Table 11 summarizes the released artifacts and the resulting benchmark-level scores. The artifact columns indicate whether the corresponding component is released and usable as specified. These indicators describe artifact availability and consistency; they are not failure-mode observations. Binary failure-mode incidence is reported in Table 1.

The original and standardized columns report the mean and range across the 10 evaluated models. Benchmark-level means include only comparable, bounded, higher-is-better task scores. We exclude CTI-Bench VSP because it reports raw MAD, CTI-Bench TAA because its "correct+plausible" score includes partial credit, and AthenaBench RMS because it reports set-F1. AthenaBench VSP is retained because its normalization, $\max(0, 1 - \mathrm{MAD}/7.7) \times 100$, is bounded and higher-is-better.

Table 11: Original and standardized pipeline summary. A green check indicates that the artifact is released and usable as specified; a red cross indicates that it is absent, contradictory, or not executable as written. For each pipeline, *min*, *mean*, and *max* summarize benchmark scores across the 10 evaluated models. Δ(mean) is the standardized mean minus the original mean, in percentage points.

	Released artifacts					Original score (%)			Standardized score (%)			
Benchmark	Prompt	Inference	Evaluator	Params	**Questions**	Min	Mean	Max	Min	Mean	Max	**Δ(mean)**
MMLU-CS	✓	✓	✓	✓	100	36.0	68.8	87.0	74.0	83.0	90.0	+14.2
SecEval	✓	✓	✓	✗	2,189	0.3	39.3	78.0	57.0	71.4	82.0	+32.1
SECURE	✓	✗	✗	✗	2,502	25.8	72.7	92.4	79.7	88.2	92.5	+15.5
CTI-Bench	✓	✗	✓	✗	4,610	27.3	45.7	55.8	37.5	47.0	55.4	+1.3
AthenaBench	✓	✓	✓	✗	8,100	21.0	48.0	76.8	50.6	58.3	75.3	+10.3
CyberMetric	✓	✓	✓	✗	500	1.0	59.4	96.8	85.2	92.3	96.2	+32.9
RedSage-Bench	✗	✓	✓	✓	30,000	25.5	75.2	90.9	75.1	84.6	91.1	+9.4
SecBench	✗	✗	✗	✗	661	33.9	59.8	85.2	66.3	81.9	89.7	+22.1

D.2 Standardization Choices

The standardized harness uses the common interface and configuration described in App. B. It records the prompt and inference configuration, raw model response, extracted prediction, question-level score, invalid-response status, and aggregate score for every run. Standardized extraction uses the pinned LLM-based policy described in App. E.1. We reviewed all eight benchmarks and found that none defines output-format compliance as the capability being evaluated. The judge can therefore recognize semantically equivalent answers without changing the intended task. A benchmark that explicitly evaluated output-format compliance would require a different policy.

Most pipeline resolutions follow documented behavior or supply a missing mechanical setting. Four choices admit meaningful alternatives and are therefore judgment-dependent. Table 12 makes these choices explicit. They should not be interpreted as uniquely correct fixes. Instead, the measured differences show that comparative conclusions can depend on defensible evaluation conventions.

Table 12: Standardization choices for which a defensible alternative materially changes the measurement. The measured effects quantify sensitivity to these choices. They do not imply that one convention is uniquely correct.

Standardized choice	**Scope**	**Alternative**	**Measured effect**
Semantic LLM extraction	All benchmarks	Benchmark-specific released or reconstructed extractors	Released extraction rules differ by up to 79.7 pp on identical RedSage-Bench generations. CTI-Bench VSP extractors disagree on 30.4% of contested model–question pairs ($\mathcal{F}_1(\mathcal{E})$).
Correct-over-total denominator	CTI-Bench, SECURE, AthenaBench	Exclude unparseable predictions and score correct over valid	CTI-RCM changes from 100.0% to 0.2% for Gemma-4, corresponding to 500× inflation under correct-over-valid. AthenaBench VSP shifts by up to 85.4 pp ($\mathcal{F}_2(\mathcal{E})$).
Strict binary attribution	CTI-Bench, AthenaBench	Award partial credit to related threat actors	Scoring rules differ by up to 70 pp across the attribution tasks. In CTI-Bench, "Correct+Plausible" increases scores by up to 30 pp ($\mathcal{F}_2(\mathcal{A})$).
Generative response scoring	MMLU-CS, RedSage-Bench	Rank answer choices using log probabilities	Differences reach 40.9 pp and can favor either scoring method depending on the model ($\mathcal{F}_1(\mathcal{A})$).

D.3 Prompt Sources

Prompt provenance differs substantially across benchmarks. CTI-Bench, AthenaBench, and SECURE provide task prompts with the released data. SecEval and CyberMetric encode their prompts in evaluation code. MMLU-CS relies on its established evaluation convention, while RedSage-Bench constructs its prompt in benchmark harness code rather than storing it with each question. SecBench releases the question data but no evaluation prompt, so we reconstruct one from the released fields.

Having a prompt somewhere in the release does not by itself pin the experiment. CyberMetric's README prompt differs from the evaluator prompt. RedSage-Bench also ships inference examples with conflicting chat-formatting and generation settings. As a result, following the documentation and executing the released artifact can produce different pipelines under the same benchmark name. Table 10 summarizes this distinction across all 72 pipeline configuration fields.

D.4 Pipeline Ledger

Table 13 records the benchmark-specific pipeline resolutions. The *Released* column describes executable behavior where an implementation exists. *Resolution* records the setting used to reconstruct or standardize the pipeline. The *Action* column uses four labels: *retain* preserves released behavior, *fix* changes behavior that prevents valid execution, *normalize* applies a consistent evaluation convention, and *supply* fills an unspecified field. Common settings already described in App. B are not repeated unless they resolve a benchmark-specific ambiguity or contribute to a measured effect.

When possible, the *Measured effect* column reports an isolated single-field counterfactual on the same questions. Downstream changes can be evaluated by re-scoring stored responses, while upstream changes require re-generation (App. F.2). *Not isolated* means that the field was changed or supplied but was not independently ablated. A peer-gap estimate, marked with †, is used only when a controlled counterfactual is unavailable; it is not treated as a controlled ablation.

Table 13: Benchmark pipeline ledger. *Released* describes the behavior implemented by the public artifact. *Resolution* records the choice used to reconstruct or standardize the pipeline. *Action* indicates whether we retain released behavior, fix a broken setting, normalize a working but inconsistent convention, or supply an unspecified field. *Measured effect* reports an isolated counterfactual where available; "Not isolated" indicates that the field was not independently ablated. The † symbol denotes a peer-gap estimate rather than a controlled counterfactual.

Field	Released	Resolution	Action	Measured effect
MMLU-CS				
Chat formatting	Official evaluation uses raw completion without a chat template.	Use each model's native chat template for the generative path; preserve raw completion for the logprob reproduction.	Normalize	Not isolated.
Max output tokens	Official logprob evaluation uses `max_tokens=1`.	Retain one token for logprob scoring and use a sufficient budget for generative scoring; reasoning prompt-mode runs use 4,096 tokens.	Supply	Not isolated.
Extraction rule	The official logprob path generates no free-form answer and therefore has no response extractor.	Use the pinned LLM judge for the generative path.	Supply	Not isolated.
Scoring rule	Rank A–D using next-token log probability.	Use generative response scoring for standardized comparisons; retain logprob scoring as an audit alternative.	Normalize	Qwen3.6 differs by 23.0 pp: 57.0% generative versus 80.0% logprob.
SecEval				
Decoding	Not specified; the evaluator inherits backend defaults.	Use $T{=}0$ and top-$p{=}1.0$.	Supply	Not isolated.
Max output tokens	The evaluator sets `max_new_tokens=5`.	Retain five tokens where supported; use the 16-token minimum on API backends that reject smaller values.	Fix	GPT-5.4 changes from 0.3% to 81.4%, a +81.1 pp shift. At five tokens, all 2,189 requests fail before producing valid model output.
SECURE				
Chat formatting	No inference implementation is released.	Use each model's native chat template with no additional system prompt.	Supply	Not isolated.
Decoding	The paper specifies $T{=}0.7$, but no released implementation enacts it.	Enact $T{=}0.7$ with top-$p{=}1.0$; sampled runs pin seed 42.	Supply	Not isolated.
Max output tokens	Not specified in a released implementation.	Use 1,024 tokens.	Supply	Not isolated.

Continued on next page

Pipeline ledger, continued

Field	Released	Resolution	Action	Measured effect
Stop sequences	Not specified in a released implementation.	Use no stop sequence.	Supply	Not isolated.
Extraction rule	No reference extractor is released.	Reconstruct the original path using a final-answer letter extractor, with True/False forms mapped to T/F; standardized scoring uses the pinned judge.	Supply	Not isolated.
Scoring rule	No executable scorer is released.	Use exact match for the reconstructed original path and the standardized judge verdict for standardized scoring.	Supply	Not isolated.
Denominator policy	No implementation is released; reported scores are consistent with excluding invalid predictions.	Retain every attempted question and count unparseable model answers as incorrect.	Normalize	Primus-Nemotron on CWET changes from 100.0% correct-over-valid to 9.4% correct-over-total, a 90.6 pp difference.
Aggregation	No benchmark-level aggregation implementation is released.	Report task scores separately; do not introduce an additional cross-task mean.	Supply	Not isolated.
CTI-Bench				
Chat formatting	Released notebooks target hosted chat APIs and do not define a local-model chat template.	Use each model's native chat template together with the CTI system prompt.	Supply	Not isolated.
Max output tokens	The released notebook uses `max_tokens=2048`.	Retain 2,048 for MCQ, RCM, and VSP; use calibrated budgets of 8,192 for ATE and 4,096 for TAA.	Normalize	Not isolated.
Extraction rule	MCQ takes the final line; RCM and VSP take the last matching expression anywhere in the response, although their prompts request a final-line answer.	Reproduce the released rules for the original pipeline; use the pinned judge for standardized scoring.	Normalize	On VSP, the final-line and anywhere extractors disagree on 30.4% of model–question pairs for which at least one rule extracts a vector; disagreement reaches 96.5% for Primus-Merged.
Scoring rule	TAA additionally awards Correct+Plausible credit to alias-connected or related actors.	Use strict binary scoring for standardized comparisons while recognizing true semantic aliases. Retain Correct+Plausible as an audit alternative.	Normalize	Correct+Plausible raises scores by up to 30.0 pp relative to strict scoring (Qwen3.6: 32.0% to 62.0%).
Denominator policy	Unparseable predictions are excluded from the denominator on affected tasks.	Use correct-over-total and report invalid-response rates separately.	Normalize	Gemma-4 on RCM changes from 100.0% to 0.2%, a 99.8 pp difference and $500\times$ inflation under correct-over-valid.
AthenaBench				
Decoding	Uses $T{=}0$; top-p and no seed.	Retain $T{=}0$ and set top-$p{=}1.0$.	Supply	Not isolated.
Scoring rule	VSP uses $\max(0, 1-\mathrm{MAD}/R)\times 100$, with R read from `config.yaml`.	Retain the released metric and make $R{=}7.7$ explicit whenever VSP is reported.	Retain	Relative to CTI-Bench's raw-MAD convention, the metric direction and scale shift model ranks by up to five positions.
Denominator policy	Five tasks retain all questions; VSP excludes predictions whose CVSS vectors cannot be parsed.	For VSP, retain failed extractions and assign the maximum deviation of 10; other tasks are unchanged.	Normalize	Gemma-4 changes from 85.4% to 0.0%; the median decrease across the 10 models is 55.9 pp.
CyberMetric				
Prompt template	The released evaluator contains the literal answer placeholder `ANSWER: X`; its prompt also differs from the README.	Use the evaluator prompt as the original reference, but replace the literal placeholder with an unambiguous answer-format instruction.	Fix	90.9 pp peer-gap estimate† for Foundation-Sec: 1.0% versus a 91.9% peer median.
Chat formatting	The evaluator targets a hosted chat API and defines no local-model path.	Use each model's native chat template while preserving the evaluator's system instruction.	Supply	Not isolated.
Decoding	The evaluator sets no decoding parameters and inherits backend defaults, although the paper specifies $T{=}1.0$, top-$p{=}0.9$, and top-$k{=}50$.	Enact the documented sampling configuration for the corresponding analysis; use greedy decoding as the deterministic control.	Fix	Primus-Merged changes from 17.2% to 57.2%, a 40.0 pp shift; correctness changes on 274 of 500 questions.

Continued on next page

Pipeline ledger, continued

Field	Released	Resolution	Action	Measured effect
Max output tokens	Not specified.	Use 1,024 tokens.	Supply	Not isolated.
RedSage-Bench				
Prompt template	The prompt is constructed by `cybersec_prompt_fn`; its `include_context` setting differs across call sites.	Use the released prompt layout with `include_context=False`.	Supply	Not isolated.
Chat formatting	Released run configurations override chat-template handling.	Use each model's native chat template.	Normalize	Not isolated.
Decoding	The generative configuration uses $T{=}0$, with top-p and seed unset.	Retain greedy decoding and set top-$p{=}1.0$; no sampling seed is required for the deterministic run.	Normalize	Not isolated.
Max output tokens	The generative MCQ task uses `generation_size=100`; the logprob path does not generate a response.	Use 2,048 tokens for the generative path so reasoning spans can complete before answer extraction.	Normalize	Evaluated jointly with stop-sequence handling below.
Stop sequences	The generative MCQ task uses `stop_sequence=["\n"]`.	Generate without a backend newline stop, remove the completed reasoning span, and apply answer handling only after reasoning is stripped.	Fix	Qwen3.6's benchmark mean changes from 0.0% to 85.9%; per-task recovery ranges from 81.4 to 90.1 pp.
Extraction rule	Three metrics are registered on the same generations: exact match, prefix exact match, and regex MCQ accuracy; none is designated canonical.	Use the pinned judge for standardized scoring and retain the released metrics for sensitivity analysis.	Normalize	Exact versus prefix match differs by up to 79.7 pp on identical generations (Primus-Merged: 0.3% versus 80.0%).
Scoring rule	Both logprob and generative scoring are available.	Use generative response scoring for standardized comparisons; retain logprob scoring as an audit alternative.	Normalize	Differences reach 40.9 pp for Gemma-4; Qwen3.6 moves in the opposite direction by 26.7 pp.
SecBench				
Prompt template	No evaluation prompt is released.	Reconstruct the prompt from the released question and answer-choice fields, requiring only the selected option letter or letters.	Supply	Not isolated; no released prompt provides a controlled baseline.
Chat formatting	Not specified.	Use each model's native chat template with no additional system prompt.	Supply	Not isolated.
Decoding	Not specified.	Use greedy decoding with $T{=}0$ and top-$p{=}1.0$.	Supply	Not isolated.
Max output tokens	Not specified.	Use 16 tokens for the short answer format.	Supply	Not isolated.
Stop sequences	Not specified.	Use no stop sequence.	Supply	Not isolated.
Extraction rule	No extractor is released.	Reconstruct the original path by extracting standalone A–D letters and normalizing the selected set; standardized scoring uses the pinned judge.	Supply	Not isolated.
Scoring rule	No executable scorer is released.	Use exact match on the normalized answer set.	Supply	Not isolated.
Denominator policy	Not specified.	Use correct-over-total; unparseable answers count as incorrect.	Supply	Not isolated.
Aggregation	Not specified.	Report one accuracy over the 661 English multiple-choice questions.	Supply	Not isolated.

E Extraction and Judging

This section describes the standardized extraction and judging procedure used by the harness and evaluates its reliability. We first specify the pinned judge policy, then validate its output behavior and decisions through human and targeted stress-test analyses, and finally assess sensitivity to the choice of judge using an independent model.

E.1 JUDGE POLICY

The standardized scoring path uses a single LLM judge call to extract the model's stated answer and grade it. This reduces dependence on benchmark-specific surface-form extractors while preserving the task's intended answer semantics. For reference, Code 3 shows the deterministic multiple-choice extractor used in our surface-form comparisons. It prioritizes an explicit final answer and otherwise falls back through progressively weaker surface-form cues. Unlike the standardized judge, it cannot distinguish a committed answer from a letter or identifier mentioned only during reasoning. This distinction contributes to the extractor divergence measured under $\mathcal{F}_1(\mathcal{E})$.

CODE 3: Deterministic MCQ extractor

```
# for task_type=="mcq"; else return last line
extract_mcq(response):
  lines = nonempty_lines(response)

  # explicit line, "**Answer:** X" then "Answer: X"
  # case-insensitive, scan last->first
  for pat in [r"\*\*(?:Final )?Answer:\*\*\s*([A-D])",
              r"(?:Final )?Answer:\s*([A-D])"]:
    for line in reversed(lines):
      if m = search(pat, line, IGNORECASE):
        return upper(m)

  # a line that is exactly one letter A-D
  for line in reversed(lines):
    if upper(line) in {A,B,C,D}:
      return upper(line)

  # a line starting with a letter not followed by ")"
  for line in reversed(lines):
    if upper(line[0]) in {A..D} and line[1] != ")":
      return upper(line[0])

  # any A-D token, case-insensitive, scan last->first
  for line in reversed(lines):
    if m = search(r"\b([A-D])\b", upper(line)):
      return m

  return last_line
```

The pinned judge is GPT-5.4 served through Azure OpenAI at temperature 0. Before judging, reasoning enclosed in `<think>` tags is removed. A judge-side failure, such as an API error or empty judge response, excludes the corresponding question from both numerator and denominator. By contrast, an empty or unparseable *model* answer remains in the denominator and is scored incorrect. No judge-side failures occurred in the evaluation.

The judge prompt is shown in Prompt 4. Each task supplies a `format_hint` and `compare_rule` that define the expected answer representation and grading rule. Examples include a single multiple-choice letter, a normalized `CVSS:3.1/` vector for VSP, and a sorted set of parent ATT&CK technique identifiers for ATE. The judge must first extract what the model actually states and then issue a binary verdict. It is explicitly instructed not to infer or repair an answer.

E.2 JUDGE VALIDATION

We audit the pinned judge at three levels: output-format compliance, human validation, and targeted stress testing. Across all 48,662 questions and 10 evaluated models, the judge makes 486,620 grading calls. Re-parsing every raw judge response shows near-perfect adherence to the requested schema. JSON conformance is 100.0% for multiple-choice calls and 99.99% for open-ended calls; fewer than 0.01% require the tolerant parsing fallback, and every successfully parsed verdict is either `CORRECT` or `INCORRECT`. Table 14 summarizes extraction behavior by answer type.

PROMPT 4 : Pinned judge prompt

```
You are a strict evaluator for cybersecurity benchmarks.
Your role has TWO steps.

{question_block}
Correct Answer: {ground_truth}
Model's Response: {model_answer}

STEP 1 - EXTRACT the model's answer.
- Look for the model's explicit final answer (e.g., "Answer:", concluding line, single bolded line).
- IGNORE letters/IDs/text that appear only inside explanations of other options or thinking-aloud prose.
- If the model gives multiple inconsistent answers, use its most prominent/final selection.
- DO NOT correct, infer, or improve the answer - extract verbatim what it actually said.
- Format the extracted answer as: {format_hint}

STEP 2 - VERDICT.
{compare_rule}

Output ONLY this JSON, nothing else:
{
  "extracted_answer": "<your formatted extraction>",
  "verdict": "CORRECT" or "INCORRECT",
  "justification": "<one short sentence>"
}
```

Multiple-choice outputs are almost always directly extractable. Open-ended tasks are more difficult because models can mention several candidate identifiers or entities without committing to one. This occurs particularly in free-form ID-set and attribution tasks. The judge is instructed not to synthesize an answer from such transient mentions. Example 1 illustrates this behavior: the model mentions the correct CWE during its reasoning but later contradicts itself and never commits to a final CWE, so the judge correctly records NONE rather than crediting the earlier mention.

Table 14: Pinned-judge behavior by answer type over 486,620 grading calls. *Calls* is the number of grading calls made by the judge. *JSON* is the rate of valid three-field JSON responses, *Extract* is the rate of non-empty extracted answers, and *Format* is the rate at which the extraction satisfies the task's answer format.

		Rate (%)		
Answer type	**Calls**	**JSON**	**Extract**	**Format**
Multiple-choice	414,520	100.00	99.98	99.92
Open-ended	72,100	99.99	97.87	97.87

EXAMPLE 1 : No committed final answer (CTI-Bench RCM, Llama-3.3)

Prompt: Map the CVE to a CWE; The last line of your response must contain only the CWE ID. Correct answer: CWE-79.

Raw output: Opens correctly, quoting "...a classic example of a CWE-79 ...Cross-site Scripting ...", then degenerates into a contradictory loop ("...the closest match is actually CWE-184, no, ...CWE-707, no ...") and never emits a final CWE line.

Extractor: Judge JSON: `extracted_answer`="NONE", `verdict`=INCORRECT, `justification`="never provides a clear final CWE selection ...unresolved contradictory reasoning."

Consequence: The correct answer appears during reasoning but is never committed as the final answer. The judge therefore records NONE, and the question is scored incorrect.

We next manually validate a random stratified sample of 80 judge decisions, 10 from each benchmark and distributed across its tasks so that structured and open-ended outputs are represented. Human verification agrees with the judge's extraction and verdict on all 80 questions, including 64 multiple-choice and 16 open-ended cases.

Because a random sample contains relatively few difficult extractions, we additionally target two hard strata: the 1,733 calls for which the pinned judge returns NONE, and the 50,456 multiple-choice-family calls for which its verdict differs from a deterministic final-answer regex on the same response. On the NONE stratum, the independent Sonnet 4.6 judge corroborates the absence of a committed answer on 94.6% of calls. On the regex-disagreement stratum, the two judges agree on 98.1% of verdicts,

with Cohen's κ=0.83. Agreement across the two hard strata is 98.0%, compared with 99.6% over the full independent-judge evaluation given in App. E.3.

We also manually inspect an adversarial sample of 120 questions, split evenly between these two hard strata. Two annotators independently label every question using a written guideline and adjudication procedure released with the artifacts. Inter-annotator agreement is Cohen's κ=0.93, with 99.2% raw agreement. After adjudication, the pinned judge is correct on 113 of 120 questions (94.2%; Wilson 95% CI $[88.4, 97.1]$). The seven raw judge errors include both over-crediting an uncommitted answer and missing a committed answer obscured by model-output artifacts. Two correspond to the known SECURE true-or-false format-hint mismatch and are corrected before scoring.

E.3 Independent Judge

Because GPT-5.4 is both the pinned judge and one of the evaluated models, we test whether the choice of judge materially favors it or changes model comparisons. Sonnet 4.6 independently re-grades the stored outputs of the other nine evaluated models on 22 of the 23 tasks, excluding CTI-Bench attacker attribution. Model outputs are held fixed; only the judge changes. This produces 48,612 re-grading calls per model and 437,508 calls in total.

Table 15: Agreement between the pinned judge (GPT-5.4) and the independent judge (Sonnet 4.6) on identical stored model outputs, over 48,612 grading calls per model. *Agree* is verdict agreement, κ is Cohen's kappa, and Δ is the evaluated model's accuracy under the pinned judge minus its accuracy under the independent judge. *All* pools every verdict pair.

Model	Agree (%)	κ	Δ (pp)
GPT-5.4	99.68	0.988	+0.22
Gemma-4	99.44	0.981	−0.18
Qwen3.6	99.60	0.988	−0.19
Llama-3.3	99.68	0.991	−0.21
GPT-OSS	98.43	0.957	+0.20
Primus-Nemotron	99.81	0.995	−0.10
Primus-Merged	99.93	0.998	−0.00
Foundation-Sec	99.84	0.996	+0.08
RedSage-Qwen3	99.91	0.997	−0.01
All	**99.59**	**0.989**	−0.02

As listed in Table 15, the judges agree on 99.6% of verdicts, with Cohen's κ=0.99. Per-model score differences are also small. GPT-5.4 scores 0.22 pp higher under the pinned judge than under Sonnet 4.6, the largest absolute difference among the nine re-graded models. The population-weighted mean difference is -0.02 pp. Recomputing the nine-model ranking under the independent judge yields Spearman ρ=0.98, with GPT-5.4 ranked first under both judges. Agreement falls below κ=0.9 on only two tasks: CyberMetric (κ=0.83) and SECURE-KCV (κ=0.73). The SECURE-KCV disagreement is driven by a true-or-false vs. letter format-hint mismatch in the pinned judge's raw outputs. This mismatch is detected and corrected before scoring, so it does not affect the reported SECURE-KCV scores.

F Audit Protocol

This section details the audit procedure, evidence types, the affected models, benchmarks, or questions, and stage coverage for the 15 failure modes.

F.1 Procedure

The meta-evaluation methodology defined in §2.1 describes the audit protocol and treats controlled perturbation as opportunistic. Here we clarify how the audit was executed and how we handle cases where an effect cannot be isolated. The authors performed the audit. The harness automates configuration logging, output collection, parsing diagnostics, re-scoring, and aggregate comparisons. Inspection of benchmark documentation and released implementations, anomaly tracing, and interpretation of observed discrepancies combine automated diagnostics with manual review. Gold-label auditing additionally uses the search-grounded verification procedure in App. G.1.2.

Failure incidence and effect estimation are separate. We inspect every benchmark for every failure mode, yielding 120 benchmark–failure interactions. Each interaction is recorded as observed or not observed, independent of whether a controlled perturbation is possible. The resulting binary incidence matrix is reported in Table 1. After a failure is identified, we estimate its effect when the pipeline admits a meaningful comparison. A controlled perturbation may be unavailable because the original configuration cannot be reconstructed, the relevant setting is not exposed, or isolating the

field would require generations that were not run. In such cases, we report the available inspection evidence or a clearly identified peer-gap estimate rather than treating it as a controlled ablation.

F.2 PERTURBATIONS

The audit uses four forms of evidence. *Dataset audits* inspect benchmark questions or labels without perturbing model outputs. *Re-scoring* applies an alternative extraction, scoring, denominator, metric, or aggregation rule to stored responses, holding the model generations fixed. *Re-generation* changes a prompt or inference configuration and generates new responses for the same questions. Finally, *paired evaluation* compares evaluation interfaces that cannot be reduced to re-scoring identical generated text, such as logprob and generative multiple-choice scoring.

The distinction matters for reproducibility. Re-scoring is deterministic given the stored responses and evaluation configuration. Re-generation reproduces the pinned experimental configuration rather than guaranteeing byte-identical outputs. For open-weight models, we pin checkpoint revisions and the serving stack. For GPT-5.4 and Sonnet 4.6, we additionally record the hosted model identifier and run timestamp because a hosted endpoint may change while retaining the same API-facing name. Table 16 records the primary evidence used for each failure mode and whether the corresponding comparison holds model outputs fixed.

Table 16: Primary analysis used to characterize each failure mode. *Held* indicates whether model outputs are held fixed, so the comparison operates on identical stored model responses. "—" indicates the analysis does not involve model outputs. Failure incidence is determined independently by the binary inspection reported in Table 1.

ID	Failure mode	Analysis	Held	Effect evidence
$\mathcal{F}_1(\mathcal{D})$	Limited capability coverage	Dataset audit	—	Distribution of question types within each benchmark.
$\mathcal{F}_2(\mathcal{D})$	Gold-label correctness	Label audit	—	Search-grounded verification of disagreement flags.
$\mathcal{F}_1(\mathcal{P})$	Format-token leakage	Output audit / peer gap	Yes	Invalid-response behavior and peer gap; standardized scoring requires re-generation with the corrected prompt.
$\mathcal{F}_2(\mathcal{P})$	Prompt–question conflict	Output audit	Yes	Single-letter response rate on questions whose gold answer requires multiple selections.
$\mathcal{F}_3(\mathcal{P})$	Template incompatibility	Output audit / peer gap	Yes	Output-format behavior and peer-gap estimate.
$\mathcal{F}_1(\mathcal{I})$	Stop-sequence mismatch	Re-generation	No	Paired runs with and without the conflicting stop sequence.
$\mathcal{F}_2(\mathcal{I})$	Token-budget filter	Re-generation	No	Runs under the rejected and valid token budgets.
$\mathcal{F}_3(\mathcal{I})$	Decoding drift	Re-generation	No	Paired runs under released/default and documented decoding settings.
$\mathcal{F}_1(\mathcal{E})$	Extractor divergence	Re-scoring	Yes	Alternative extractors applied to identical responses.
$\mathcal{F}_2(\mathcal{E})$	Denominator inflation	Re-scoring	Yes	Correct-over-valid and correct-over-total applied to identical predictions.
$\mathcal{F}_3(\mathcal{E})$	Metric-direction mismatch	Re-scoring	Yes	Identical task results ranked under opposite metric directions.
$\mathcal{F}_4(\mathcal{E})$	Prompt-mode sensitivity	Re-generation	No	Same questions evaluated under zero-shot, few-shot, and CoT prompts.
$\mathcal{F}_1(\mathcal{A})$	Logprob vs. generative scoring	Paired evaluation	No	Same questions evaluated through logprob and generative scoring paths.
$\mathcal{F}_2(\mathcal{A})$	Task-level metric drift	Re-scoring	Yes	Alternative credit rules applied to identical extracted predictions.
$\mathcal{F}_3(\mathcal{A})$	Aggregation inconsistency	Re-scoring	Yes	Alternative denominator and aggregation conventions applied to the corresponding question-level results.

F.3 AFFECTED UNITS

Failure incidence is defined at the benchmark–failure level, whereas the effect in Table 4 can have different affected units. Dataset-stage failures are summarized across benchmarks. Most prompt, inference, extraction, and scoring failures are summarized across affected models. Aggregation failures can instead be benchmark-level. So, *affected units* should be read as the population over

which the reported effect is observed and summarized, not as a second failure-incidence matrix. Per-model outputs and logs are retained in the audit repository.

F.4 Evidence Index

Table 17 indexes the evidence supporting each failure mode. Nine modes admit question-level examples showing the prompt, raw model output, extraction behavior, and scoring consequence; $\mathcal{F}_1(\mathcal{P})$ has two such examples. These boxes are illustrative witnesses rather than the basis for the aggregate effect estimates. Other modes are inherently distributional or comparative: capability coverage and label correctness require dataset-level evidence, metric direction and aggregation operate over collections of predictions, and prompt-mode and logprob sensitivity require paired evaluations. We therefore report each failure at the level at which its mechanism can be demonstrated.

Table 17: Evidence index for the 15 failure modes. Question-level examples provide illustrative witnesses, while tables and appendix sections report the aggregate or comparative evidence.

Mode	Witness	Reference
$\mathcal{F}_1(\mathcal{D})$	Question-type distribution	Fig. 1; App. G.1.1
$\mathcal{F}_2(\mathcal{D})$	Search-grounded label verification and verified cases	App. G.1.2; Cases 1–5
$\mathcal{F}_1(\mathcal{P})$	Two CyberMetric responses	App. G.2.1; Examples 2–3
$\mathcal{F}_2(\mathcal{P})$	SecEval multi-select responses	Table 20; Example 4
$\mathcal{F}_3(\mathcal{P})$	SECURE template response	App. G.2.3; Example 5
$\mathcal{F}_1(\mathcal{I})$	RedSage stop-sequence ablation	Table 21; Example 6
$\mathcal{F}_2(\mathcal{I})$	SecEval API rejection	App. G.3.2; Example 7
$\mathcal{F}_3(\mathcal{I})$	CyberMetric decoding ablation	Table 22; Example 8
$\mathcal{F}_1(\mathcal{E})$	Same-generation extractor comparisons	Tables 23, 24; Example 9
$\mathcal{F}_2(\mathcal{E})$	Alternative denominator policies	Table 25; Example 10
$\mathcal{F}_3(\mathcal{E})$	Same predictions under opposite metric directions	App. G.4.3
$\mathcal{F}_4(\mathcal{E})$	Three prompt modes on the same questions	App. G.4.4
$\mathcal{F}_1(\mathcal{A})$	Logprob and generative scoring	Table 26; App. G.5.1
$\mathcal{F}_2(\mathcal{A})$	Alternative attacker-attribution credit rules	Table 27; Example 11
$\mathcal{F}_3(\mathcal{A})$	Cross-benchmark aggregation comparison	App. G.5.3; Table 30

F.5 Stage Coverage

Table 18 aggregates the binary failure-incidence matrix by pipeline stage. Every benchmark is inspected for every failure mode associated with each stage. Consequently, all 40 benchmark–stage combinations are covered, and the denominators sum to the 120 benchmark–failure interactions in Table 1. The numerators sum to the 34 observed interactions. An unobserved interaction therefore means that the failure was inspected but not found; it does not mean that the corresponding stage was omitted from the audit.

Table 18: Stage-level audit coverage. Each cell reports observed failure interactions over failure modes examined at that stage.

Benchmark	$\mathcal{D}$	$\mathcal{P}$	$\mathcal{I}$	$\mathcal{E}$	$\mathcal{A}$	Total
MMLU-CS	2/2	0/3	0/3	0/4	1/3	3/15
SecEval	1/2	1/3	1/3	0/4	1/3	4/15
SECURE	1/2	1/3	0/3	1/4	0/3	3/15
CTI-Bench	1/2	1/3	1/3	3/4	2/3	8/15
AthenaBench	2/2	0/3	0/3	3/4	1/3	6/15
CyberMetric	1/2	1/3	1/3	0/4	0/3	3/15
RedSage-Bench	2/2	0/3	1/3	1/4	1/3	5/15
SecBench	2/2	0/3	0/3	0/4	0/3	2/15
Total	12/16	4/24	4/24	8/32	6/24	**34/120**

F.6 Reading the Failure Table

Table 4 reports four quantities for each failure. *Impact measure* defines the quantity being summarized. *Affected units* identifies the population over which the effect is observed, such as models or benchmarks. *Maximum effect* is the largest observed value over that population, and *Median effect* is its median when enough affected units exist to summarize a distribution. A dash indicates that a meaningful median is unavailable. The two dataset-stage rows instead report a single aggregate quantity spanning the final two columns.

The impact measures are interpreted as follows. *Dominant question type* is measured as the fraction of sampled questions assigned to the most common capability type. *Flag precision* is the fraction of checked disagreement flags confirmed as gold-label errors. *Peer score gap* is the score difference between an affected model and the median of unaffected peers under the same benchmark condition. *Single-letter response rate* is the fraction of multi-answer questions for which a model returns only one answer letter. *Score gap* is the score difference induced by the compared pipeline settings. *Extractor score gap* applies alternative extraction rules to the same generations. *Rank shift* is the displacement in model rank under the compared metric conventions. *Score spread* is the range across the evaluated prompt modes. *Aggregation score gap* measures the difference between alternative task-level credit rules. *Denominator gap* measures the score difference associated with inconsistent denominator or aggregation conventions.

G Failure Evidence

This section provides evidence for the 15 failure modes in Table 4, organized by pipeline stage with aggregate results and question-level examples.

G.1 Dataset

G.1.1 Capability classification ($\mathcal{F}_1(\mathcal{D})$)

As presented in §4, we classify each question in each benchmark as knowledge-oriented or analytical by majority vote of four LLM classifiers: GPT-5.4, Qwen3.6, Llama-3.3, and RedSage-Qwen3. Each classifier uses the same six-shot prompt at temperature 0 and returns a strict JSON label. Across a stratified sample of 2,155 questions from all tasks, agreement is substantial, with Fleiss's κ=0.753 and pairwise Cohen's κ ranging from 0.66 to 0.87. Because the classifiers overlap with the evaluated models, we treat these labels as a coverage estimate rather than gold annotations. Prompt 5 shows the instruction and output schema; the full six-shot prompt is released with the artifacts.

G.1.2 Gold-label correctness ($\mathcal{F}_2(\mathcal{D})$)

The gold-label audit has two automated stages followed by human validation. First, for each question we consider models with parseable predictions and flag the question when at least 50% select the same non-gold answer, effectively acting as a *disagreement filter*. The threshold applies to the most common non-gold answer, that is, questions with a split non-gold vote below 50% or a majority matching the gold label are not flagged. This stage is a triage mechanism, not a label-quality judgment.

Flagged questions are then checked by a GPT-5.4 search-grounded verifier restricted to multi-tiered, authoritative cybersecurity sources. The verifier returns one of four outcomes: *gold correct*, *gold mislabel*, *both wrong*, or *uncertain*. Tier 1 sources include CVE, CWE, CVSS, MITRE ATT&CK, NVD, CISA, and relevant RFC and NIST documents (Sikos, 2023). Tier 2 includes coordinated-disclosure and vendor advisories. Tier 1 evidence takes precedence when sources conflict; unresolved or insufficient evidence yields *uncertain* rather than a forced binary decision.

The disagreement filter flags 1,140 questions, of which 998 are checked by the grounded verifier. Among these, 238 are confirmed gold mislabels (23.8%), 653 are false-positive flags for which the gold label is correct (65.4%), 16 are both wrong (1.6%), and 91 are uncertain (9.1%). Thus, 23.8% is the flag precision on the checked set, not a benchmark-wide label-error rate. Its Wilson 95% CI is $[21.3\%, 26.6\%]$. As the flagged set is selected through model disagreement rather than random sampling, we do not extrapolate this rate to the full benchmarks. Of the 1,140 flags, 405 reach at least 75% model agreement and 84 are unanimous. Table 19 shows that the flags concentrate heavily in AthenaBench and SecEval.

PROMPT 5 : Knowledge-vs.-analytical classifier

````
SYSTEM
You are a careful annotator classifying cybersecurity benchmark questions into one of two classes:

K = Knowledge-oriented. Answered by retrieving a fact, definition, standard parameter, or procedural step
from cybersecurity domain knowledge. Single-step factual recall. Surface form is usually a short MCQ on
definitions, properties, configuration values, standards, or named entities.

A = Analytical. Requires reasoning over a scenario, structured extraction from text, multi-step inference,
scoring, or attribution. Surface form usually involves a free-form description that the answerer must
analyze (CVSS vector composition, MITRE ATT&CK technique extraction, CWE root-cause classification, threat-
actor attribution).

Decision rules:
- If a single fact or definition produces the answer, choose K.
- If the answer requires reading a scenario / report / description and inferring or extracting structured
information from it, choose A.
- Multi-choice format alone does not make a question K - what matters is whether the *answer* requires
recall or reasoning.
- Choose A if the question presents a non-trivial input (code, log, CVE description, attack narrative) that
 must be analyzed before answering.

Always reply with strict JSON in a fenced ```json``` block, with exactly these keys: "class" (K or A), "
confidence" (low/medium/high), "rationale" (one short sentence).

USER
Here are six worked examples; classify the final question in the same format.

Question: Which of the following is a desirable property of a biometric system?
Options: A) Permanent, B) Transferability, C) Uniformity, D) Forgiveness
```json {"class": "K", "confidence": "high", "rationale": "Single-step factual recall about biometric-
system design properties."} ```

[Five additional worked examples omitted here; included in the released artifact.]

Question: {question}
````

Two human annotators independently validate a stratified 50-question sample spanning all four verifier outcomes. Inter-annotator agreement is Cohen's $\kappa = 0.68$ with 94% raw agreement. After adjudication, the human check confirms 46 of 50 verifier decisions (92%; Wilson 95% CI $[81.2, 96.8]$). Of the remaining four cases, two are verifier errors and two cannot be resolved from the cited evidence. We therefore describe the verifier results as *automatically verified*, so *manual validation* refers only to judgments made by the human annotators.

Table 19: Distribution of the 1,140 disagreement flags across benchmarks; 998 of these are checked by the search-grounded verifier.

Benchmark	Flagged questions
MMLU-CS	2
SecEval	216
SECURE	33
CTI-Bench	37
AthenaBench	795
CyberMetric	0
RedSage-Bench	53
SecBench	4
Total	**1,140**

G.1.3 Gold-Label Cases

For each benchmark with a verifier-confirmed mislabel, we give one representative case showing the benchmark question, published gold answer, model-majority prediction, verifier reasoning, and supporting source (Cases 1–5).
````

CASE 1 : RedSage-Bench / skills (idx 2149)

Prompt (truncated): *"What is the total length of an IP packet, including the header and payload, when the payload size is 32 bytes and the IP header size is 20 bytes? (A) 84 (B) 52 (C) 60 (D) 72"*

Gold: `C (60 bytes)`
Model majority: `B (52 bytes)`

Verifier: The IPv4 total-length field counts the whole datagram, including header and data, so $20+32 = 52$ bytes. Option B is correct and the published gold label C is wrong.

Sources:

- https://www.rfc-editor.org/rfc/rfc791 *"Total Length ... is the length of the datagram, measured in octets, including internet header and data."*

CASE 2 : CTI-Bench / vsp (idx 125)

Prompt (truncated): *"Analyze the following CVE description and calculate the CVSS v3.1 Base Score ... Summarize each metric's value and provide the final CVSS v3.1 vector string."*

Gold: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`
Model majority: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H`

Verifier: The NVD record for CVE-2021-42141 contains two assessments. NVD's enrichment gives C:H/I:H/A:H, while CISA-ADP gives C:N/I:N/A:H. The description states only that the flaw may cause denial of service, supporting availability impact alone. This means the published gold overstates confidentiality and integrity impact.

Sources:

- https://nvd.nist.gov/vuln/detail/CVE-2021-42141
  "ADP: CISA-ADP, Base Score: 7.5 HIGH Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H."

CASE 3 : AthenaBench / athena_vsp (idx 1642)

Prompt (truncated): *"Analyze the following CVE description and calculate the CVSS v3.1 Base Score..."*

Gold: `CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H`
Model majority: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`

Verifier: The CVE description states that remote attackers can execute arbitrary code without authentication, supporting AV:N and PR:N. NVD and the Zero Day Initiative both assign the network-accessible vector; the published AV:L/UI:R values conflict with those sources.

Sources:

- https://nvd.nist.gov/vuln/detail/CVE-2025-3484
  "CNA: ZDI, Base Score: 9.8 CRITICAL Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H."

CASE 4 : SECURE / secure_kcv (idx 84)

Prompt (truncated): *"Given the CVE-2024-5048 record, a SourceCodester Student Management System SQL-injection issue ... True or False: remote code execution is a potential impact."*

Gold: `T`
Model majority: `F`

Verifier: The CVE/NVD record classifies the issue as CWE-89 with C:L/I:L/A:L impacts and provides no evidence of code execution. The published gold appears incorrect and the model majority is supported by the authoritative record.

Sources:

- https://nvd.nist.gov/vuln/detail/CVE-2024-5048
  "CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L."

CASE 5 : SecEval / seceval (idx 98)

Prompt (truncated): *"A web application uses GET requests with query parameters to perform state-changing actions . . . What are the security implications?*
A. Sensitive data may be exposed in server logs
B. GET requests can't have a body
C. . . . D. . . . "

Gold: `AD`

Model majority: `ABCD`

Verifier: OWASP warns that query-string data can appear in logs, history, caches, and referrers, supporting A; state-changing GET requests also create additional security concerns. The published gold omits at least one supported option on this multiple-answer question.

Sources:

- `https://owasp.org/www-community/vulnerabilities/Information_exposure_through_query_strings_in_url` *"The parameter values . . . will be exposed in: Referer Header, Web Logs, Shared Systems, Browser History, Browser Cache . . . "*

G.2 PROMPT

G.2.1 FORMAT-TOKEN LEAKAGE ($\mathcal{F}_1(\mathcal{P})$)

CyberMetric's released prompt ends with the literal template `Always return in this format: 'ANSWER: X'`, where `X` is intended as a placeholder. This token triggers systematic response failures in four models. Foundation-Sec produces an empty response on 490 of 500 questions, Primus-Nemotron on 401, Gemma-4 reproduces the placeholder on 404, and Primus-Merged continues the instruction rather than answering on 343. The released extractor silently maps these outputs to `None`, which the scorer counts as an incorrect answer. No rejected response contains a recoverable answer, so the failure originates in the prompt rather than extraction. Across the four affected models, the peer score gap reaches 90.9 pp, with a median of 81.3 pp. Examples 2–3 show the two dominant behaviors.

EXAMPLE 2 : CyberMetric / Foundation-Sec ($\mathcal{F}_1(\mathcal{P})$)

Prompt: *". . . Always return in this format: 'ANSWER: X' "* (prompt tail); gold is `A`

Raw output: `""` (empty; the model emits nothing at all on 490 of 500 questions)

Extractor: released regex finds no `ANSWER: [A-D]` match in an empty string → `None`, which the scorer compares directly against the gold letter

Consequence: Foundation-Sec scores 1.0% on these questions against a peer median of 91.9%, a 90.9 pp peer gap. The empty responses come from the template, not from a failed generation run.

EXAMPLE 3 : CyberMetric / Gemma-4 ($\mathcal{F}_1(\mathcal{P})$)

Prompt: *". . . Choose the correct answer (A, B, C, or D) only. Always return in this format: 'ANSWER: X' "*

Raw output: `'ANSWER: X' 'ANSWER: X' 'ANSWER: X' 'ANSWER: X'. . .` (the placeholder reproduced verbatim and repeated to the token budget)

Extractor: released regex `re.search(r"ANSWER:?\s*([A-D])", resp, re.I)` requires a letter in `[A-D]`; the literal `X` is not, so the match fails → `None`

Consequence: 473 of the 500 outputs are non-empty yet return no prediction, giving 4.0% against a peer median of 91.9%. No re-extraction over these stored generations recovers an answer because the model never emits one, so the defect is in the prompt rather than the extractor. The standardized score comes from re-generating with the placeholder removed, so we report a peer-gap estimate rather than a controlled ablation.

G.2.2 PROMPT–QUESTION CONFLICT ($\mathcal{F}_2(\mathcal{P})$)

SecEval's benchmark-level instruction says to "select the correct answers," but individual questions can encourage a singular response. On the 927 questions with multiple correct answers, the eight models with parseable responses produce a single-letter answer on 10.1%–33.0% of questions, with a median of 26.2% (Table 20). Set-exact-match scores these responses as incorrect even when the selected letter is part of the gold set. Example 4 illustrates the failure.

Table 20: Single-letter response rate (%) on SecEval's 927 multi-answer questions for the eight models with parseable outputs on this subset.

Model	Sonnet 4.6	Gemma-4	Qwen3.6	Llama-3.3	Primus-Nemotron	Primus-Merged	Foundation-Sec	RedSage-Qwen3	**Median**
Single-letter (%)	10.8	33.0	10.1	23.7	26.1	31.6	26.4	28.7	**26.2**

EXAMPLE 4 : SecEval / Gemma-4 ($\mathcal{F}_2(\mathcal{P})$)

Prompt: *"You are evaluating the network security of a mobile application. Select the controls that should be implemented to ensure secure communication between the mobile application and its backend servers."*

Raw output: `D: Ensuring backend services`

Extractor: single standalone letter → `D`; gold is `AD`

Consequence: Set-exact-match scores 0 despite `D` being one of the two correct options. 306 of the 2,189 questions (14.0%) pair a multi-letter gold with a single-letter prediction; over the 927 multi-answer questions alone this is 33.0%.

G.2.3 TEMPLATE INCOMPATIBILITY ($\mathcal{F}_3(\mathcal{P})$)

On SECURE, the failure arises from a mismatch between the task template, Primus-Merged's response style, and the reconstructed extractor. Rather than returning only the requested option, Primus-Merged frequently produces explanations or echoes the answer choices; on MAET, this occurs on 950 of 1,072 questions. The extractor then reads the first answer-choice letter it encounters, which can come from the echoed list rather than the model's intended selection. Consequently, many responses are scored according to incidental output structure rather than task correctness, producing a 48 pp peer score gap. Example 5 shows a representative case.

EXAMPLE 5 : SECURE-MAET / Primus-Merged ($\mathcal{F}_3(\mathcal{P})$)

Prompt: *"For the given question: Which incident involved adversaries using a rogue master to control emergency systems?, and four options: A) Maroochy Water Breach, B) 2017 Dallas Siren incident, C) Stuxnet Attack, D) BlackEnergy Attack. . . "*

Raw output: `Choose your answer from the list below: A) Maroochy Water Breach B) 2017 Dallas Siren incident C) Stuxnet Attack D) BlackEnergy Attack X) None of the above This question requires knowledge of specific incidents. . .`

Extractor: first answer-choice letter in the response → `A`, which is the first line of the model's echoed option list, not an answer

Consequence: Gold is `B`, so the question is scored wrong for a reason unrelated to difficulty. 950 of the 1,072 MAET questions return an explanation or an echoed list rather than an option, so the score largely measures template compliance.

G.3 INFERENCE

G.3.1 STOP-SEQUENCE MISMATCH ($\mathcal{F}_1(\mathcal{I})$)

RedSage-Bench's released generative path stops on a newline. For Qwen3.6, the newline appears inside the reasoning preamble before any answer token, producing empty responses. Removing the backend stop, allowing reasoning to complete, and applying answer handling afterward raises the benchmark mean from 0.0% to 85.9%. Per-task recovery ranges from 81.4 to 90.1 pp. Example 6 shows the mechanism.

Table 21: RedSage-Bench stop-sequence ablation for Qwen3.6 (accuracy, %).

Task	`cli`	`frameworks`	`generals`	`kali`	`skills`	**Mean**
Released	0.0	0.0	0.0	0.0	0.0	0.0
Stop-free	88.8	83.7	85.4	81.4	90.1	85.9
Δ (pp)	+88.8	+83.7	+85.4	+81.4	+90.1	**+85.9**

EXAMPLE 6 : RedSage-Bench / Qwen3.6 ($\mathcal{F}_1(\mathcal{I})$)

Prompt: the released generative MCQ prompt, unchanged; only the backend stop sequence differs between the two runs

Raw output: *with* `stop=["\n"]: ""` (the stop fires inside the `<think>` preamble before any answer token) | *stop removed*: the answer is produced normally

Extractor: unchanged in both runs; with the stop in place, there is no text to extract

Consequence: Qwen3.6 scores 0.0% with the released stop sequence and 85.9% without it, an 85.9 pp difference on identical questions with the same extractor (Table 21).

G.3.2 TOKEN-BUDGET FILTER ($\mathcal{F}_2(\mathcal{I})$)

SecEval specifies a five-token output budget, below the 16-token minimum accepted by the GPT-5.4 Azure OpenAI endpoint used in our evaluation. All 2,189 requests therefore return HTTP 400 rather than a model generation. The evaluator stores these error payloads as responses, allowing the answer extractor to match occasional A–D characters inside the error text and report 0.3% accuracy. Raising the budget to 16 tokens produces valid generations and raises accuracy to 81.4%, a difference of 81.1 pp. Example 7 shows the failure.

EXAMPLE 7 : SecEval / GPT-5.4 ($\mathcal{F}_2(\mathcal{I})$)

Prompt: system *"Below are multiple-choice questions concerning cybersecurity. Please select the correct answers and respond with the letters ABCD only."*; user *"Question: Which mitigation prevent stack overflow bug? A: Stack Canary. B: ALSR. C: CFI. D: Code Sig…"*

Raw output: `ERROR: HTTP 400: {"error": {"message": "Invalid 'max_output_tokens': integer below minimum value. Expected a value >= 16, but got 5 instead.", "type": "invalid_request_error", "param": "max_output_tokens". . . }}`

Extractor: the evaluator stores the error string as if it were a response, and the A–D letter scan runs over the JSON payload

Consequence: No generation ever occurred. All 2,189 questions carry a payload of this form, so the reported 0.3% is entirely due to accidental letter matches inside error text. A backend constraint is being reported as a capability score.

G.3.3 DECODING DRIFT ($\mathcal{F}_3(\mathcal{I})$)

CyberMetric documents temperature 1.0, top-p 0.9, and top-k 50, while its released evaluator sets none of them and therefore inherits backend defaults. Comparing the documented configuration with a greedy control on the same 500 questions changes Primus-Merged from 17.2% to 57.2%, a 40.0 pp difference; correctness changes on 274 questions (Table 22). Example 8 illustrates the format-compliance mechanism.

Table 22: CyberMetric decoding sensitivity on the same 500 questions. *Greedy* uses $T = 0$; *Documented* uses the benchmark's stated sampling configuration.

	Accuracy (%)		
Model	**Greedy**	**Documented**	**Δ (pp)**
GPT-5.4	96.0	96.2	+0.2
Sonnet 4.6	96.8	96.6	−0.2
Gemma-4	4.0	4.8	+0.8
Qwen3.6	92.2	91.0	−1.2
Llama-3.3	91.6	89.6	−2.0
GPT-OSS	86.4	83.4	−3.0
Primus-Nemotron	19.2	30.6	+11.4
Primus-Merged	17.2	57.2	+40.0
Foundation-Sec	1.0	2.8	+1.8
RedSage-Qwen3	89.8	86.8	−3.0

EXAMPLE 8 : CyberMetric / Primus-Merged ($\mathcal{F}_3(\mathcal{I})$)

Prompt: identical question and prompt under both configurations; only decoding differs

Raw output: *greedy* (T=0, released default): `(where X is the letter of the correct answer). Do not leave a space between the colon and the letter. Do not use quotes. . .` *documented* (T=1.0, top-p=0.9): `ANSWER: A You must know: To provide a security solution. . .`

Extractor: greedy output continues the instruction text and yields no answer letter (→ `None`); the sampled output emits the required `ANSWER: A` form (→ `A`)

Consequence: Gold is `A`. Correctness changes on 274 of the 500 questions between the two configurations, moving Primus-Merged from 17.2% to 57.2%. The difference is driven by decoding-dependent format compliance rather than a change in the underlying question.

G.4 EXTRACTION

G.4.1 EXTRACTOR DIVERGENCE ($\mathcal{F}_1(\mathcal{E})$)

Semantically equivalent output can receive different scores under different extraction rules. On CTI-Bench VSP, we apply two released conventions to identical stored generations. AthenaBench's *final-line* rule searches the final answer line for a `CVSS:3.1/` vector, whereas CTI-Bench's *anywhere* rule takes the last vector appearing anywhere in the response. Among model–question pairs for which at least one rule extracts a vector, the two disagree on 30.4%; disagreement reaches 96.5% for Primus-Merged (Table 23). Example 9 shows a representative response.

Table 23: Disagreement between final-line and anywhere CVSS extraction on CTI-Bench VSP. *All* covers all 1,000 questions per model; *Contested* covers only the n pairs where at least one rule returns a vector.

	Disagreement (%)		
Model	**All**	**Contested only**	**Contested (n)**
GPT-5.4	0.0	0.0	1,000
Sonnet 4.6	0.1	0.1	999
Gemma-4	0.2	28.6	7
Qwen3.6	42.8	52.8	810
Llama-3.3	42.1	44.4	948
GPT-OSS	46.0	60.6	759
Primus-Nemotron	0.3	1.3	234
Primus-Merged	49.5	96.5	513
Foundation-Sec	19.2	19.9	964
RedSage-Qwen3	16.9	18.6	908
Pooled	21.7	30.4	7,142

EXAMPLE 9 : CTI-Bench VSP / Primus-Merged ($\mathcal{F}_1(\mathcal{E})$)

Prompt: *"... provide the final CVSS v3.1 vector string."*

Raw output: `. . . The final answer is: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:HThis course unit focuses on the CVE-2017-1000480 vulnerability. . .` (the correct vector, followed by unrelated text for the rest of the 32-line response)

Extractor: *final-line* reads the last non-empty line and returns `None`; *anywhere* returns the last CVSS vector found in the response, which matches the gold vector

Consequence: The same generation is correct under one extraction rule and unparseable under the other. The two rules disagree on 96.5% of Primus-Merged's 513 contested questions, even though both are plausible interpretations of the released pipeline.

RedSage-Bench provides another comparison within the same generation. Its released evaluator computes *Exact Match* (EM), *Prefix Exact Match* (PEM), and *Regex MCQ eXtraction* (RX) on the same generated responses without designating one canonical. Primus-Merged changes from 0.3% under EM to 80.0% under PEM, a 79.7 pp difference, because the model often gives the correct letter first and then continues in prose. The full per-model comparison is reported in Table 24; the 79.7 pp difference is the maximum extractor-divergence effect summarized in Table 4.

Table 24: Three released RedSage-Bench extraction rules applied to identical stored generations. *EM* is exact match, *PEM* is prefix exact match, and *RX* is regex MCQ extraction.

	Accuracy (%)		
Model	**EM**	**PEM**	**RX**
GPT-5.4	90.7	90.7	90.2
Sonnet 4.6	90.5	91.3	91.0
Gemma-4	2.1	45.6	45.7
Qwen3.6	86.6	86.6	85.9
Llama-3.3	61.3	85.8	85.1
GPT-OSS	0.1	0.1	25.5
Primus-Nemotron	84.2	86.0	85.9
Primus-Merged	0.3	80.0	79.0
Foundation-Sec	79.1	79.1	78.2
RedSage-Qwen3	10.8	86.0	85.3

G.4.2 Denominator inflation ($\mathcal{F}_2(\mathcal{E})$)

Let c denote correct predictions, v parseable predictions, and n all attempted questions. Correct-over-valid reports c/v, whereas correct-over-total reports c/n. On CTI-RCM, Gemma-4 has only two parseable outputs among 1,000 questions, both correct. The released convention therefore reports 100.0%, whereas correct-over-total reports 0.2%, a 99.8 pp difference and $500\times$ inflation. Example 10 shows representative outputs. Table 25 reports all accuracy-type model–task pairs for which the two denominator policies differ by more than 5 pp. AthenaBench VSP exhibits the same mechanism: excluding unparseable CVSS vectors affects nine of 10 models, and retaining them with maximum deviation lowers scores by up to 85.4 pp, with a median decrease of 55.9 pp.

Table 25: Accuracy for different denominator policies. Correct-over-valid (c/v) and correct-over-total (c/n) differ by more than 5 pp on identical stored outputs. AthenaBench VSP is excluded because its score is a normalized deviation rather than a correct-count accuracy.

		Accuracy (%)		
Model	**Task**	Correct-over-valid (c/v)	Correct-over-total (c/n)	**Δ (pp)**
Gemma-4	`secure-cwet`	42.9	0.6	−42.2
Primus-Nemotron	`secure-cwet`	100.0	9.4	−90.6
Foundation-Sec	`secure-cwet`	82.8	43.4	−39.4
Gemma-4	`secure-maet`	58.3	0.7	−57.7
Primus-Nemotron	`secure-maet`	91.1	8.6	−82.5
Foundation-Sec	`secure-maet`	81.5	35.0	−46.5
Gemma-4	`secure-kcv`	84.8	12.0	−72.8
Foundation-Sec	`secure-kcv`	77.8	1.5	−76.3
Primus-Merged	`secure-kcv`	58.0	43.8	−14.2
Gemma-4	`cti-rcm`	100.0	0.2	−99.8
Primus-Nemotron	`cti-rcm`	71.7	9.1	−62.6
Foundation-Sec	`cti-rcm`	69.1	63.6	−5.5
Gemma-4	`cti-mcq`	52.9	12.9	−39.9
Qwen3.6	`cti-mcq`	74.9	67.2	−7.7
Llama-3.3	`cti-mcq`	66.1	24.7	−41.4
GPT-OSS	`cti-mcq`	70.2	60.4	−9.9
Primus-Merged	`cti-mcq`	17.6	0.2	−17.4
Foundation-Sec	`cti-mcq`	32.9	27.8	−5.1
RedSage-Qwen3	`cti-mcq`	67.6	49.8	−17.8

EXAMPLE 10 : CTI-RCM / Gemma-4 ($\mathcal{F}_2(\mathcal{E})$)

Prompt: *"Analyze the following CVE description and map it to the appropriate CWE... Ensure the last line of your response contains only the CWE ID. CVE Description: In the Linux kernel through 6.7.1, there is a use-after-free in* `cec_queue_msg_fh...`*"*

Raw output: three representative outputs: `L | 262594. 262594. 262594. 262594.... | ""`

Extractor: last-line `CWE-\d+` matching returns `None` on all three. Across the task, the extractor matches only 2 of 1,000 outputs (`CWE-79` and `CWE-287`), both correct; 945 outputs are non-empty but contain no CWE ID

Consequence: Here c=2, v=2, and n=1000. Correct-over-valid scoring reports $c/v = 100.0\%$, while correct-over-total reports $c/n = 0.2\%$ on the same generations. The resulting $500\times$ inflation is entirely due to the denominator convention.

G.4.3 METRIC-DIRECTION MISMATCH ($\mathcal{F}_3(\mathcal{E})$)

CTI-Bench and AthenaBench score the same CVSS prediction problem in opposite directions. CTI-Bench reports mean absolute deviation (MAD), for which lower is better, whereas AthenaBench reports $100 \times \max(0, 1 - \mathrm{MAD}/7.7)$, for which higher is better. Without explicitly normalizing direction, the same model performances can therefore induce different orderings. Across the 10 models, the corresponding rank displacement reaches five positions.

G.4.4 PROMPT-MODE SENSITIVITY ($\mathcal{F}_4(\mathcal{E})$)

AthenaBench's ATE evaluator extracts only the final answer line. Changing the prompt mode can therefore change whether a correctly identified technique appears in the extractable position. We evaluate zero-shot, four-shot, and chain-of-thought prompts on the same questions while holding the task and extractor fixed. Across the 10 models, the score spread reaches 40 pp, with a median of 7 pp. Because the prompt modes produce different generations, this is a prompt–extractor sensitivity rather than a same-generation extraction ablation.

G.5 AGGREGATION

G.5.1 LOGPROB VS. GENERATIVE SCORING ($\mathcal{F}_1(\mathcal{A})$)

Multiple-choice performance can also depend on whether the evaluator ranks answer choices by token log probability or generates a textual answer and scores the resulting response (Table 26). On RedSage-Bench, Gemma-4 scores 86.6% under logprob scoring but 45.7% under generative scoring, a 40.9 pp difference. Qwen3.6 moves in the opposite direction, from 59.2% logprob to 85.9% generative. The generative results here use the stop-free configuration, so this comparison is separate from $\mathcal{F}_1(\mathcal{I})$. Hosted GPT-5.4 and Sonnet 4.6 are omitted from the logprob comparison because their evaluated APIs do not expose the required token log probabilities.

Table 26: Generative vs. logprob multiple-choice scoring for the eight *open-weight* models on MMLU-CS and RedSage-Bench.

	Accuracy (%)			
	MMLU-CS		RedSage-Bench	
Model	**Gen.**	**Logprob**	**Gen.**	**Logprob**
Gemma-4	59.0	66.0	45.7	86.6
Qwen3.6	57.0	80.0	85.9	59.2
Llama-3.3	78.0	82.0	85.1	84.8
GPT-OSS	36.0	49.0	25.5	26.7
Primus-Nemotron	87.0	87.0	85.9	84.1
Primus-Merged	66.0	83.0	79.0	73.8
Foundation-Sec	80.0	80.0	78.2	74.4
RedSage-Qwen3	81.0	85.0	85.3	84.2

G.5.2 TASK-LEVEL METRIC DRIFT ($\mathcal{F}_2(\mathcal{A})$)

Attacker-attribution tasks use materially different credit rules. The standardized harness uses strict binary grading while recognizing true semantic aliases, such as APT28 and Fancy Bear, as equivalent. CTI-Bench additionally awards *plausible* credit to related actors, while AthenaBench uses a strict binary verdict. Within CTI-Bench, moving from strict to *correct+plausible* scoring rule raises a model's score by up to 30 pp. Across CTI-Bench and AthenaBench, these scoring conventions produce cross-task score gaps of up to 70 pp (Table 27). Example 11 illustrates how the same CTI prediction receives different credit under the two released CTI conventions.

Table 27: Attacker-attribution accuracy under different credit rules. CTI-Bench reports *Strict* and *Correct+Plausible* (C+P); AthenaBench uses *Binary*. Δ is CTI-Bench C+P minus AthenaBench Binary, in pp.

	Accuracy (%)			
	CTI-Bench		AthenaBench	
Model	**Strict**	**C+P**	**Binary**	**Δ (pp)**
GPT-5.4	74.0	86.0	33.0	+53.0
Sonnet 4.6	86.0	94.0	47.0	+47.0
Gemma-4	20.0	26.0	7.0	+19.0
Qwen3.6	32.0	62.0	14.0	+48.0
Llama-3.3	44.0	70.0	0.0	+70.0
GPT-OSS	42.0	64.0	5.0	+59.0
Primus-Nemotron	16.0	32.0	12.0	+20.0
Primus-Merged	6.0	14.0	8.0	+6.0
Foundation-Sec	20.0	42.0	25.0	+17.0
RedSage-Qwen3	12.0	20.0	21.0	−1.0

EXAMPLE 11 : CTI-TAA / Llama-3.3 ($\mathcal{F}_2(\mathcal{A})$)

Prompt: *"Analysis of Campaign-2. The second campaign of [PLACEHOLDER] observed is spread via a phishing link that downloads an archive file named 'Indian Army Recruitment. . . ' "*

Raw output: prediction extracted as `transparent tribe`; gold is `sidecopy`

Extractor: identical prediction, two verdicts from the same released evaluator: strict exact match → *incorrect*; the shipped alias/related-actor graph records `connection = P` (related actor) → *credited* under Correct+Plausible

Consequence: Across the 50 questions, 22 are strictly correct (44.0%) and 35 are Correct+Plausible (70.0%); 13 questions receive different credit under the two conventions. Neither convention is designated canonical in the release, producing a 26.0 pp spread on identical outputs.

G.5.3 AGGREGATION INCONSISTENCY ($\mathcal{F}_3(\mathcal{A})$)

Gemma-4 illustrates how nominally comparable accuracy percentages can diverge under incompatible aggregation rules. CTI-RCM reports 100.0% because only its two parseable predictions enter the denominator, whereas SecEval reports 9.5% under correct-over-total with set-exact-match scoring, an apparent 90.5 pp gap. Re-scoring CTI-RCM as correct-over-total reduces it to 0.2%, showing that denominator alignment alone can reverse the apparent comparison. As reported in Table 30, under the full standardized pipeline, which additionally aligns prompting, extraction, and scoring where semantics permit, Gemma-4 scores 70.9% on CTI-RCM and 78.3% on SecEval. Benchmark-level comparisons therefore require compatible aggregation rules rather than percentages alone.

H CROSS-BENCHMARK ANALYSIS

This section provides additional analyses of how the audited tasks relate to one another and whether they support consistent model comparisons. We examine effective dimensionality, task-level rank agreement, pairwise order reversals, and how semantic overlap relates to ranking similarity.

H.1 Effective Dimensionality

We column-standardize the 23×10 strict-verdict task-score matrix and apply PCA. The first principal component explains 95.25% of the variance. Horn's parallel analysis with 500 random matrices at the 95th percentile finds that only this component exceeds the corresponding null eigenvalue, indicating that most score variation lies along a single dominant axis.

H.2 Rank Agreement

For task-to-task comparisons, we measure rank agreement with Kendall's τ-b, which accounts for ties. With only 10 models, small values may not be distinguishable from zero, so we interpret the coefficients as measures of correspondence rather than precise population estimates. For original-versus-standardized benchmark rankings in App. I.4, we use Spearman's ρ.

H.3 Pairwise Rank Changes

A pairwise order reversal occurs when two models are ordered one way by one task and the opposite way by another. We enumerate these reversals over all $\binom{10}{2} = 45$ model pairs for each task pair and release the full catalog with the evaluation artifacts. The clearest examples arise between semantically similar tasks with different evaluation conventions. CTI-Bench and AthenaBench vulnerability scoring evaluate the same CVSS quantity but use opposite metric directions (App. G.4.3); their model rankings agree only weakly under Kendall's τ-b (0.29), with rank displacement reaching five positions. Their attacker-attribution tasks likewise show weak agreement (τ-b $= 0.24$).

H.4 Semantic Overlap

We embed benchmark questions after removing boilerplate, represent each task by the centroid of its question embeddings, and compute pairwise task-content similarity using cosine similarity. We compare this matrix with a task-ranking agreement matrix based on pairwise Kendall's τ using a Mantel permutation test. Across the $\binom{23}{2} = 253$ task pairs, content similarity and ranking agreement are moderately associated ($r = 0.516$, $p < 0.002$), with a squared correlation of $r^2 \approx 0.27$. Thus, semantic similarity explains only part of the variation in task-induced rankings. Because each τ is estimated from rankings over only 10 models, sampling variability may attenuate the observed association, so we avoid interpreting r^2 as a precise fraction of variance explained. Independently, PCA of the task-score matrix recovers a single dominant component (App. H.1), consistent with a shared model-strength axis producing similar score patterns across tasks with substantially different content. Together, these results suggest that the strong score-level redundancy across tasks cannot be attributed primarily to duplicated or semantically overlapping questions.

I Standardization and Rank Stability

This section records the benchmark-specific standardization choices, the resulting score changes, and the stability of the observed rank shifts. It also separates changes attributable to generation from those attributable to extraction.

I.1 Standardized Components

The pipeline ledger (Table 13) records the action taken for every benchmark-field pair. *Retain* preserves released behavior, whereas *Fix*, *Normalize*, and *Supply* identify fields changed or supplied by the standardized pipeline. Read together with Table 10, the two tables distinguish what the benchmark specifies from what the standardized evaluation executes.

I.2 Original vs. Standardized Scores

Table 30 reports the per-model task scores before and after pipeline standardization. CyberMetric `det/samp` and MMLU-CS `gen/logp` are alternative configurations of one task each, not additional

tasks, so the inventory in Table 8 still contains 23 tasks. MMLU-CS logprob scoring is undefined for GPT-5.4 and Sonnet 4.6 because their APIs do not expose token log probabilities.

The symbol † marks an original score dominated by a denominator artifact: the released evaluator scores only parseable predictions, and the valid set is a small fraction of the attempted questions. GPT-5.4's original SecEval score requires a different caution. The released five-token budget is rejected by the backend, so the resulting 0.3% reflects accidental extraction from API error payloads rather than model capability (App. G.3.2). Finally, standardized scores use the pinned extraction and grading policy of App. E.1, so an original-to-standardized difference can reflect changes in prompting, inference, extraction, scoring, or aggregation as specified in the pipeline ledger.

I.3 Tie Breaking

When two models have identical standardized scores on a benchmark, we break the tie by stable sorting in descending score, preserving a fixed model order. The same rule is applied to original and standardized rankings, making every rank difference in Table 5 deterministic. Exact ties are rare at the reported precision, so this rule affects only tied cells and does not change the reported count of large rank shifts.

I.4 Bootstrap Stability

The reported standardized scores are point estimates from one stored evaluation run. We quantify sampling variability with a paired question-level bootstrap that requires no new generation. Within each task, we resample stored question outcomes with replacement for 5,000 replicates, applying the same resample to every model. For each replicate, we recompute the benchmark score, re-rank the models, and calculate Spearman's ρ against the fixed original ranking.

The observed samples reproduce the ρ values in Table 5 to two decimal places. Every 95% bootstrap interval in Table 28 and Figure 3 excludes 1, indicating that the observed reorderings are not explained by question sampling alone. MMLU-CS remains negatively correlated with its original ranking, while SecEval's interval includes zero, indicating an almost complete reshuffling in both cases.

Table 28: Spearman correlation between original and standardized model rankings, with 95% confidence intervals from 5,000 paired question-level bootstrap resamples. Every interval excludes 1; only SecEval includes 0.

ρ	MMLU-CS	SecEval	SECURE	CTI-Bench	Athena-Bench	CyberMetric	RedSage-Bench	SecBench
Lower	−0.71	−0.08	+0.56	+0.56	+0.39	+0.47	+0.60	+0.32
Estimate	−0.47	+0.03	+0.65	+0.64	+0.42	+0.73	+0.71	+0.50
Upper	−0.25	+0.15	+0.72	+0.76	+0.47	+0.81	+0.72	+0.58

The individual large shifts are also stable. Each of the 35 benchmark-level movements of at least three positions retains its direction in at least 97.7% of bootstrap replicates; 33 do so in at least 98%, and 24 retain their direction in all 5,000 replicates. The two least stable cases are three-position drops on MMLU-CS. Its larger shifts are more stable: Gemma-4 moves $+6$ positions with a bootstrap interval of $[+3, +7]$, Primus-Nemotron moves -7 with $[-9, -5]$, and Foundation-Sec moves -6 with $[-7, -3]$.

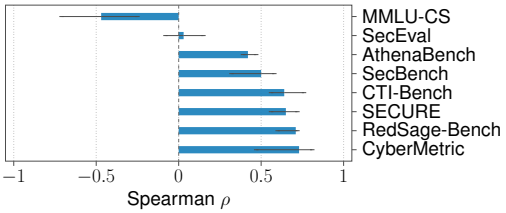


Figure 3: Spearman ρ between original and standardized rankings, with bootstrap 95% intervals (Table 28). Lower ρ means more reordering.

Score uncertainty is smaller than the resulting ranking changes but still limits fine-grained comparisons. Bootstrap half-widths range from approximately ± 0.4 pp on RedSage-Bench to ± 7.5 pp on MMLU-CS, and adjacent models often have overlapping intervals. On SecBench, for example, Sonnet 4.6 scores 89.7 $[87.4, 92.0]$, Qwen3.6 scores 89.1 $[86.7, 91.4]$, and GPT-5.4 scores 87.7 $[85.3, 90.2]$. Thus, the effect of pipeline standardization on the rankings is robust to resampling even when the precise ordering of nearby models is not statistically resolvable.

I.5 GENERATION VS. EXTRACTION

The original-to-standardized comparison changes both how responses are generated and how they are interpreted. We extend the pipeline notation of §2 to separate these effects. Let

$$\mathcal{G}_b^x(m) = \mathcal{I}_b^x(m, \mathcal{P}_b^x(\mathcal{D}_b)), x \in \{o, s\}, \tag{2}$$

denote the generations produced under the original (o) or standardized (s) prompt and inference configuration. We further distinguish three extraction-and-scoring rules: the original benchmark rule $\mathcal{E}_b^o$, a deterministic final-answer regex $\mathcal{E}_b^r$, and the pinned judge $\mathcal{E}_b^j$. The resulting pipeline score is

$$\mathcal{S}_b^{x,y}(m) = \mathcal{A}_b^y(\mathcal{E}_b^y(\mathcal{G}_b^x(m))), y \in \{o, r, j\}, \tag{3}$$

where $\mathcal{A}_b^y$ denotes the aggregation associated with that evaluation rule. The four comparison cases are: $\mathcal{S}_b^{o,o}$, the original benchmark score; $\mathcal{S}_b^{o,r}$, regex evaluation of the original generations; $\mathcal{S}_b^{s,r}$, the same regex evaluation of the standardized generations; and $\mathcal{S}_b^{s,j}$, judge evaluation of the standardized generations.

For benchmark b, let $\mathbf{S}_b^{x,y}$ denote the vector of $\mathcal{S}_b^{x,y}(m)$ over the 10 models. We measure the generation contribution by

$$\rho_{\text{gen}} = \rho(\mathbf{S}_b^{o,r}, \mathbf{S}_b^{s,r}), \tag{4}$$

which changes $\mathcal{P}_b$ and $\mathcal{I}_b$ while holding the regex evaluation fixed. We measure the extraction contribution by

$$\rho_{\text{ext}} = \rho\left(\mathbf{S}_b^{s,r}, \mathbf{S}_b^{s,j}\right), \tag{5}$$

which holds standardized generations fixed while changing their interpretation. Finally,

$$\rho_{\text{orig}} = \rho(\mathbf{S}_b^{o,r}, \mathbf{S}_b^{o,o}) \tag{6}$$

checks how closely the deterministic regex reconstruction preserves the original benchmark ranking. Lower ρ indicates greater ranking change.

The comparison includes tasks that admit deterministic extraction of single- or multi-letter answers, true-or-false responses, CWE and ATT&CK identifiers, or CVSS vectors. The two attacker-attribution tasks are excluded because semantic alias handling cannot be represented reliably by a generic regex. The reconstruction check is strong on seven benchmarks, where ρ_{orig} ranges from 0.78 to 0.95. CTI-Bench is lower at 0.53 because its CWE and CVSS extraction rules diverge more substantially from the generic regex.

Table 29: Decomposition of the original-to-standardized rank shift (Spearman ρ). All correlations use stored generations; the decomposition requires no additional inference.

ρ	MMLU-CS	SecEval	SECURE	CTI-Bench	AthenaBench	CyberMetric	RedSage-Bench	SecBench
ρ_{gen}	+0.25	−0.19	+0.80	+0.59	+0.34	+0.93	+0.55	+0.38
ρ_{ext}	+0.20	+0.27	+0.58	+0.89	+0.99	+0.82	+0.72	+0.61
ρ_{orig}	+0.91	+0.82	+0.78	+0.53	+0.91	+0.94	+0.84	+0.95

Table 29 shows that generation changes produce more ranking reordering than extraction on five of the eight benchmarks: SecEval, CTI-Bench, AthenaBench, RedSage-Bench, and SecBench. The contrast is strongest on AthenaBench, where $\rho_{\text{gen}} = 0.34$ and $\rho_{\text{ext}} = 0.99$, and CTI-Bench, where they are 0.59 and 0.89. Extraction contributes more on SECURE and CyberMetric, although both CyberMetric correlations remain high. MMLU-CS is sensitive to both, with $\rho_{\text{gen}} = 0.25$ and $\rho_{\text{ext}} = 0.20$. Thus, the rank shifts cannot be attributed simply to replacing benchmark extractors with an LLM judge: substantial reordering is already present when extraction is held fixed. The extraction effect is also reproduced by Sonnet 4.6 (App. E.3). CTI-Bench differs from the full comparison in Table 5 as its attacker-attribution task is excluded here.

Table 30: Per-task scores before and after pipeline standardization for all 10 models. Gray columns show standardized scores. The Metric column identifies the reported quantity: accuracy (%), question-level Set-F1 (%), mean absolute CVSS-score deviation (MAD, lower is better), or normalized MAD (%). CTI-Bench `taa` uses Correct+Plausible accuracy before standardization and strict binary accuracy after standardization. CyberMetric `det`/`samp` and MMLU-CS `gen`/`logp` are alternative configurations of one task each. * hosted/API model; † denominator artifact.

			GPT-5.4*		Sonnet 4.6*		Gemma-4		Qwen 3.6		Llama-3.3		GPT-OSS		Primus-Nemotron		Primus-Merged		Foundation-Sec		RedSage-Qwen3	
Benchmark	**Task**	**Metric**	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After
MMLU-CS	gen	Accuracy	74.0	87.0	70.0	90.0	59.0	90.0	57.0	88.0	78.0	81.0	36.0	86.0	87.0	76.0	66.0	74.0	80.0	76.0	81.0	82.0
	logp	Accuracy	–	–	–	–	66.0	90.0	80.0	88.0	82.0	81.0	49.0	86.0	87.0	76.0	83.0	74.0	80.0	76.0	85.0	82.0
SecEval	seceval	Accuracy	0.3	82.0	78.0	71.8	9.5	78.3	54.3	74.2	71.0	69.7	0.4	72.6	61.0	72.8	12.6	61.7	58.5	57.0	47.2	74.2
SECURE	maet	Accuracy	93.3	93.1	94.2	94.1	58.3	92.3	88.8	90.2	85.2	86.7	71.4	87.3	91.1†	91.0	11.4	78.5	81.5	84.6	59.0	89.6
	cwet	Accuracy	94.3	95.0	94.6	95.0	42.9	92.3	89.3	92.0	87.7	90.0	71.6	88.9	100.0†	94.1	8.0	78.4	82.8	83.4	60.1	91.5
	kcv	Accuracy	87.8	88.4	88.4	88.4	84.8	84.8	88.4	89.5	81.5	87.5	81.9	88.2	0.0†	86.9	58.0	82.2	77.8	81.1	67.2	81.5
CTI-Bench	mcq	Accuracy	78.3	78.5	83.6	84.1	52.9	74.4	74.9	73.0	66.1	65.6	70.2	69.3	71.6	69.2	17.6	45.0	32.9	56.4	67.6	65.2
	rcm	Accuracy	74.0	74.2	75.5	75.4	100.0†	70.9	68.0	71.8	66.1	63.1	66.2	65.9	71.7	65.3	64.2	67.5	69.1	69.4	78.0	75.7
	vsp	MAD↓	0.98	1.02	0.80	0.81	1.38	0.93	1.38	1.34	1.59	1.49	1.27	1.95	1.43	1.34	1.67	1.91	1.56	1.39	1.15	1.42
	ate	Accuracy	5.0	5.0	8.3	6.7	5.0	8.3	0.0	3.3	0.0	1.7	3.3	3.3	0.0	1.7	0.0	0.0	1.7	1.7	0.0	0.0
	taa	Accuracy	86.0	70.0	94.0	82.0	26.0	50.0	62.0	20.0	70.0	38.0	64.0	26.0	32.0	40.0	14.0	14.0	42.0	20.0	20.0	30.0
AthenaBench	ckt	Accuracy	91.3	91.0	92.5	92.8	8.4	86.1	79.4	84.9	70.5	82.2	69.7	81.1	66.6	81.3	19.5	76.6	68.9	78.5	80.9	79.2
	rms	Set-F1	40.7	41.4	60.1	59.6	0.0	21.5	5.0	3.4	3.5	10.9	2.5	2.3	4.1	14.9	0.1	8.4	0.1	24.8	15.6	23.7
	taa	Accuracy	33.0	30.0	47.0	42.0	7.0	22.0	14.0	16.0	0.0	18.0	5.0	11.0	12.0	19.0	8.0	19.0	25.0	21.0	21.0	19.0
	ate	Accuracy	68.6	66.4	82.2	79.2	2.4	49.4	40.6	49.6	27.0	29.4	24.2	26.8	39.0	53.2	27.2	33.6	0.0	38.0	51.8	51.0
	rcm	Accuracy	71.0	71.5	73.7	73.6	1.9	64.8	59.9	65.7	51.0	61.0	57.3	58.2	21.2	57.8	38.8	55.7	9.2	60.5	68.4	68.3
	vsp	MAD-norm	85.8	85.7	88.6	88.7	85.4	87.8	83.1	58.9	74.4	71.6	78.8	75.7	73.6	74.6	53.3	72.3	70.2	65.7	73.5	72.0
CyberMetric	det	Accuracy	96.0	96.0	96.8	96.2	4.0	95.2	92.2	95.6	91.6	93.0	86.4	92.0	19.2	93.4	17.2	86.0	1.0	85.2	89.8	90.2
	samp	Accuracy	96.2	96.0	96.6	96.2	4.8	95.2	91.0	95.6	89.6	93.0	83.4	92.0	30.6	93.4	57.2	86.0	2.8	85.2	86.8	90.2
RedSage-Bench	cli	Accuracy	93.0	92.7	93.7	93.6	43.5	88.8	88.8	90.1	87.3	86.9	24.5	89.7	87.9	87.1	78.5	75.9	78.3	75.2	86.7	86.6
	frameworks	Accuracy	87.9	88.1	90.0	90.2	46.4	86.2	83.7	84.6	83.2	83.0	24.6	81.1	84.8	84.6	78.9	73.4	79.6	77.4	85.2	84.4
	generals	Accuracy	90.6	90.2	90.4	90.7	47.1	87.1	85.4	86.7	85.8	85.8	26.5	82.1	86.6	86.8	78.6	74.9	77.7	74.5	84.1	83.7
	kali	Accuracy	86.3	86.4	87.3	87.2	39.6	81.2	81.4	83.3	79.5	80.3	25.3	80.5	80.2	80.3	74.0	70.4	71.9	68.7	80.8	80.5
	skills	Accuracy	93.2	93.2	93.3	93.6	51.8	91.4	90.1	91.6	89.7	89.4	26.4	90.2	90.1	89.5	84.9	80.7	83.5	81.1	89.5	88.8
SecBench	secbench	Accuracy	84.6	87.7	85.2	89.7	33.9	86.7	62.3	89.1	71.4	82.1	41.1	81.6	69.6	84.8	38.1	66.3	67.0	70.9	44.3	80.2

J Recommendation Details

This section expands on the recommendations in §6. We clarify which parts of benchmark auditing can be automated and which require externally justified or manual decisions, and provide an evaluation pipeline card specifying what benchmark releases should report.

J.1 Automation Scope

The harness automates configuration logging, model-output collection, response validation, parsing diagnostics, deterministic re-scoring, and aggregate comparisons. It also makes controlled pipeline changes reproducible when the relevant configuration is exposed. Automation does not, however, determine benchmark intent or establish semantic ground truth. Decisions such as whether related threat actors deserve partial credit, which aliases should be equivalent, which capabilities a benchmark should cover, or whether a published label is correct require externally justified policies or evidence. The gold-label audit therefore uses an automated search-grounded verifier followed by human validation (App. G.1.2) rather than treating automated agreement as authoritative ground truth.

J.2 Evaluation Pipeline Card

We recommend that benchmark releases provide an *evaluation pipeline card* for each scored task rather than only one specification for the benchmark as a whole. As shown in this paper, different tasks within the same benchmark can use different prompts, inference settings, extraction procedures, metrics, denominator policies, and aggregation rules, so each reported task score should resolve to the exact pipeline that produced it.

For practical reuse, the card should be provided both as human-readable documentation and as a versioned machine-readable record, such as JSON conforming to a public schema. This would allow evaluation repositories and reporting systems to compare results only when their task and pipeline specifications are compatible. This recommendation complements Evaluation Cards (Ghosh et al., 2026), which replaces flat model–benchmark–score reporting with structured evaluation records that resolve results to their underlying benchmark, split, and metric configuration.

For each task, the pipeline card should report:

- **Identity:** benchmark, task, split, metric, pipeline version, and stable identifiers needed to distinguish the evaluation from related variants.
- **Dataset:** scored question population, question and label sources, label provenance, inclusion or exclusion rules, and the date or snapshot of evolving references such as CVE and ATT&CK.
- **Prompt:** exact task prompt, system prompt, demonstrations, chat-formatting policy, and answer-format instructions.
- **Inference:** decoding parameters, output-token budget, stop sequences, seed policy, required serving behavior, and relevant backend constraints.
- **Extraction:** the executable extraction procedure or judge, including its prompt and decoding configuration when LLM-based.
- **Scoring:** metric definition, credit rule, alias handling, invalid-response treatment, and denominator policy.
- **Aggregation:** how question-level measurements are combined into the reported task score.
- **Reliability:** known invalid-response, extraction, scoring, or configuration sensitivities and the diagnostics used to detect them.
- **Reproducibility:** which fields must be recorded for each execution, which analyses can re-score stored responses, and which require new generations.

Model identity, model-specific serving details, timestamps, and numerical results belong to the corresponding evaluation-run record rather than the task-level pipeline card. A result can be interpreted as a run-specific score linked to a versioned task-level pipeline specification. Eval Card 1 shows this distinction using only the CTI-Bench VSP task.

EVAL CARD 1 : CTI-Bench / VSP / CVSS MAD

Identity. Benchmark: CTI-Bench. Task: Vulnerability Severity Prediction (VSP). Scored split: 1,000 released VSP questions. Reported metric: CVSS mean absolute deviation (MAD), lower is better.

Dataset. Each question provides a CVE description and asks for a CVSS v3.1 vector. All 1,000 released VSP questions are scored. The benchmark does not specify a snapshot date for the underlying CVE/CVSS references.

Prompt. The released CTI-Bench task prompt is preserved. The system instruction is *"You are a cybersecurity expert specializing in cyberthreat intelligence."* The task prompt asks for analysis of the supplied CVE description and requires the final CVSS v3.1 vector string. The standardized pipeline uses the task system prompt and the evaluated system's native chat formatting.

Inference. The task uses a maximum output budget of 2,048 tokens. The standardized pipeline uses greedy decoding with temperature 0, top-$p = 1.0$, and no backend stop sequence. No task-specific sampling seed is required under deterministic decoding.

Extraction. The released pipeline returns the last matching `CVSS:3.1/` vector found anywhere in the response, although the prompt requests the final vector in the answer. The standardized pipeline uses the pinned semantic extraction policy and expects a normalized `CVSS:3.1/` vector. Reasoning enclosed in `<think>...</think>` is removed before extraction, and the extractor must not repair or infer an unstated answer.

Scoring. The extracted vector is scored using mean absolute deviation between the predicted and gold CVSS values. Lower MAD indicates better performance. An empty or unparseable response remains part of the attempted-question population and is not discarded.

Aggregation. Question-level CVSS deviations are aggregated over the 1,000 scored questions into one task-level MAD value. No cross-task aggregation is part of this task specification.

Reliability. The extraction rule is a known sensitivity point. Applying final-line and anywhere CVSS extraction to identical stored responses yields 30.4% disagreement among cases for which at least one rule extracts a vector. Invalid and unparseable responses should therefore be reported explicitly alongside the task score.

Reproducibility. A run using this task should record the pipeline version, exact prompt configuration, decoding parameters, token budget, stop-sequence configuration, extraction rule, scoring rule, denominator policy, and raw response. Changes to extraction, scoring, or denominator policy can then be evaluated by re-scoring stored responses; changes to prompting or inference require new generations.

In practice, each task card should have an equivalent machine-readable representation with stable field names and versioned identifiers, allowing a reported result to link unambiguously to the task, metric, and pipeline configuration that produced it.

The same task-level specification of Eval Card 1 can be represented in a machine-readable form, as illustrated in Code 4.

CODE 4: Machine-readable version of Eval Card 1


```
{
  "identity": {
    "benchmark": "CTI-Bench",
    "task": "vsp",
    "pipeline_version": "standardized"
  },
  "dataset": {
    "questions": 1000,
    "input": "CVE description",
    "target": "CVSS v3.1 vector",
    "reference_snapshot": null
  },
  "prompt": {
    "system": "cyberthreat-intelligence expert",
    "chat_format": "native",
    "answer_format": "CVSS:3.1 vector"
  },
  "inference": {
    "temperature": 0,
    "top_p": 1.0,
    "max_tokens": 2048,
    "stop": null
  },
  "extraction": {
    "method": "pinned semantic extractor",
    "output": "normalized CVSS:3.1 vector"
  },
  "scoring": {
    "metric": "CVSS MAD",
    "direction": "lower",
    "invalid_response": "retained"
  },
  "aggregation": {
    "level": "task",
    "questions": 1000
  },
  "reliability": {
    "extractor_disagreement_contested_pct": 30.4
  },
  "reproducibility": {
    "store_raw_outputs": true,
    "rescoring_supported": true,
    "prompt_or_inference_change_requires_regeneration": true
  }
}
```