ATOMS TO AGENTS: EFFICIENT PROXIES FOR CYBERSECURITY AGENTIC EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

End-to-end benchmarks for cybersecurity large language model (LLM) agents are costly to run: they require interactive environments, tool execution, and multi-turn reasoning, yet ultimately provide only a coarse measure of overall task success. This makes them expensive for repeated model comparison and provides limited visibility into the underlying operations that contribute to success or failure. We introduce ATOMS TO AGENTS, a framework that decomposes agent trajectories into recurring atomic tasks, organizes them into a trajectory-derived capability taxonomy, and converts each into an isolated evaluation with an independently verifiable answer. By posing each task from a fixed trajectory state, the framework prevents upstream errors from propagating and avoids full multi-turn rollouts. The resulting tasks assess both operation execution and, where applicable, action selection in context, localizing model weaknesses to specific capabilities. On a panel of 13 models evaluated on ExCyTIn and CTI-REALM—the two benchmarks represented in bank construction—atomic-task scores correlate with full agentic performance at Spearman $\rho \approx 0.9$. Compared with uniformly sampled benchmark subsets that achieve the same Spearman correlation with the full-benchmark model ranking, our best observed 25-item proxy uses $76\times$ fewer tokens on ExCyTIn and $96\times$ fewer on CTI-REALM. Transfer to a new workflow is weaker when relevant capabilities are missing from the evaluation set, but the framework's taxonomy helps identify these gaps and guide targeted expansion, hence improving ranking agreement. These findings support efficient model comparison and capability diagnosis while highlighting the need for coverage-aware maintenance and end-to-end validation.

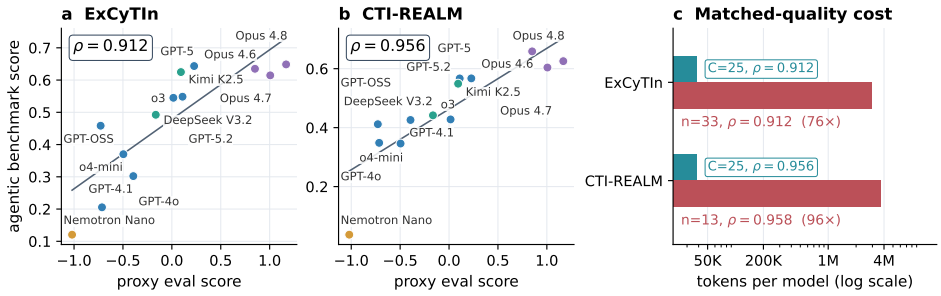


Figure 1: **Model-ranking accuracy and cost of our proxy versus agentic benchmark subsampling.** (a–b) Scores predicted by our 25-question proxy versus full benchmark scores. (c) Matching the proxy's rank correlations requires, on average, 33 ExCyTIn questions or 13 CTI-REALM tasks, using $76\times$ and $96\times$ as many tokens, respectively (both measured). The proxy is the best observed 25-question selection in our target-blind budget sweep. Table 1 reports the fixed-budget comparison; Appendix C.4 details the cost accounting.

1 INTRODUCTION

Large language models (LLMs) are increasingly being developed as autonomous agents for complex cybersecurity tasks, including penetration testing, vulnerability discovery and exploitation (Deng

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.

et al., 2024; Wan et al., 2024; Shao et al., 2024; Zhang et al., 2025a; Gioacchini et al., 2024), as well as security engineering and defensive operations (Lee et al., 2026; Zhang et al., 2025b; Chakraborty et al., 2026). Unlike static question answering, realistic agent benchmarks require models to interact with executable environments, invoke tools, observe the consequences of their actions, and make sequences of dependent decisions (Yang et al., 2023; Liu et al., 2024; Song et al., 2026). They measure downstream behavior directly, but running them repeatedly during model development demands far more infrastructure, interaction, and inference than static evaluation (Kapoor et al., 2026): PACE reports that a single evaluation on SWE-bench or GAIA can take days and thousands of dollars (Song et al., 2026), a cost paid again for every checkpoint, variant, and ablation. Benchmark compression answers this for static suites (Polo et al., 2024; Li & Xiong, 2025), but agents additionally require long-horizon inference and environment orchestration.

More importantly, this expense often yields a coarse signal: many benchmarks compress an entire trajectory into a single outcome, while aggregate success rates reveal little about an agent's progress (Ma et al., 2024). Cybersecurity tasks comprise dependent operations: exploring the environment, identifying evidence, selecting and executing tools, interpreting observations, and deciding what to do next. End-to-end success requires completing and coordinating subgoals (Deng et al., 2024; Zhang et al., 2025a; Chakraborty et al., 2026), drawing on long-horizon reasoning, decision making, and instruction following. AgentBench identifies weaknesses in these capabilities as major sources of failure (Liu et al., 2024). Cybersecurity benchmarks have introduced intermediate subtasks, milestones, and trajectory-based rewards to capture partial progress (Zhang et al., 2025a; Gioacchini et al., 2024; Chakraborty et al., 2026). However, these approaches still require end-to-end execution, and early failures can propagate through the trajectory, obscuring an agent's ability to perform later operations independently.

A less expensive alternative is static capability evaluation, with cybersecurity benchmarks covering broad knowledge (Tihanyi et al., 2024), threat-intelligence tasks such as attribution and ATT&CK extraction (Alam et al., 2024), and secure coding and offensive capability (Bhatt et al., 2024; Wan et al., 2024). However, strong knowledge-benchmark performance does not necessarily translate into effective agentic execution (Deng et al., 2024; Liu et al., 2024), and realistic tasks remain challenging even for models with substantial security knowledge (Zhang et al., 2025a; Shao et al., 2024; Lee et al., 2026). We argue that this mismatch partly reflects a difference in *abstraction level*: *static capability benchmarks assess prerequisites for agentic behavior, whereas end-to-end success depends on applying those capabilities through concrete workflow operations.* A model may possess relevant knowledge and reasoning skills yet fail to extract salient evidence from a tool response, choose the next action, or verify an intermediate objective. This distinction is particularly relevant to cybersecurity, where tasks combine the long-context and cross-artifact reasoning demands of software engineering (Jimenez et al., 2024) with reasoning about vulnerabilities, concealed evidence, and adversarial activity (Deng et al., 2024; Zhang et al., 2025a; Chakraborty et al., 2026).

This motivates evaluating atomic tasks independently to reduce execution cost and isolate local capabilities from upstream failures. Their scores could also serve as *proxies* for end-to-end performance. PACE selects roughly 100 instances from existing non-agentic benchmarks and learns a mapping to agentic performance, achieving Spearman correlation above 0.8 at under 1% of the cost (Song et al., 2026). However, its source benchmarks do not explicitly assess capabilities within cybersecurity workflows; in our evaluation, PACE-selected generic 100-item proxies achieve a mean correlation of 0.553 across the two cybersecurity benchmarks (Table 1). This raises the question: *what must a proxy measure to predict success in these workflows?*

We introduce ATOMS TO AGENTS, a framework that identifies recurring operations in recorded cybersecurity agent trajectories and turns them into standalone evaluation tasks, which we call *atomic tasks*. Unlike generic capability benchmarks, atomic tasks directly reflect cybersecurity workflows; unlike cybersecurity knowledge benchmarks, they test whether a model can apply its knowledge to interpret evidence, select actions, and verify outcomes within those workflows. The framework organizes these operations into a trajectory-derived capability taxonomy and constructs isolated evaluations with independently verifiable answers, using performance on atomic tasks to predict end-to-end agent performance.

Our Contributions. We claim three contributions in this paper:

- We present ATOMS TO AGENTS, a framework that decomposes trajectories of agents executing cybersecurity benchmark tasks into atomic operations, organizes these operations into an empir-

ical capability taxonomy, and constructs evaluation items with independently verified answers, enabling inexpensive assessment of capabilities used in security workflows.

- **Efficient prediction of agentic performance.** Across 13 models and two in-domain cybersecurity benchmarks Chakraborty et al. (2026); Wu et al. (2026), atomic-task evaluation achieves rank correlations of approximately 0.9 with end-to-end performance. From the 2,094-item bank we construct from agent trajectories, selecting and evaluating just 25 items per model at the best observed budget reduces token cost by 76–96× relative to agentic benchmark subsets with matched ranking accuracy.
- **Taxonomy-guided expansion to new workflows.** The taxonomy identifies capabilities missing from the original bank and guides construction of additional evaluation items. For (DeepTempo, 2026), an exploratory extension adds 315 items to the reusable bank, increasing its size from 2,094 to 2,409. Holding the per-model evaluation budget at 100 selected items, target-blind rank correlation increases from 0.55 to 0.78.

2 METHOD

ATOMS TO AGENTS constructs an inexpensive proxy for end-to-end cybersecurity agent evaluation using smaller, independently testable tasks. We first formalize the prediction problem and define atomic operations and evaluation items (Section 2.1). We then decompose recorded agent trajectories into recurring operations, organize them into a capability taxonomy, and construct self-contained atomic tasks with independently verifiable answers (Section 2.2). Finally, we evaluate models on the resulting atomic-task bank and test how well these inexpensive scores recover model rankings on full agentic benchmarks (Section 2.3).

2.1 FORMALIZATION

Let $\mathcal{M} = \{m_1, \ldots, m_{|\mathcal{M}|}\}$ be a set of language models, and B a target agentic benchmark with tasks $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$, an execution environment, and a scorer. Running model m on task t yields a trajectory $\tau_{m,t} = (a_1, o_1, \ldots, a_{L_{m,t}}, o_{L_{m,t}})$ of actions and observations, where $L_{m,t}$ is the run length, and a score $y_{m,t} \in [0, 1]$. These scores form the target matrix $Y \in [0, 1]^{|\mathcal{M}| \times |\mathcal{T}|}$, with $Y[m, k] = y_{m,t_k}$. Model m's mean target score is $\bar{y}_m = |\mathcal{T}|^{-1} \sum_{k=1}^{|\mathcal{T}|} y_{m,t_k}$.

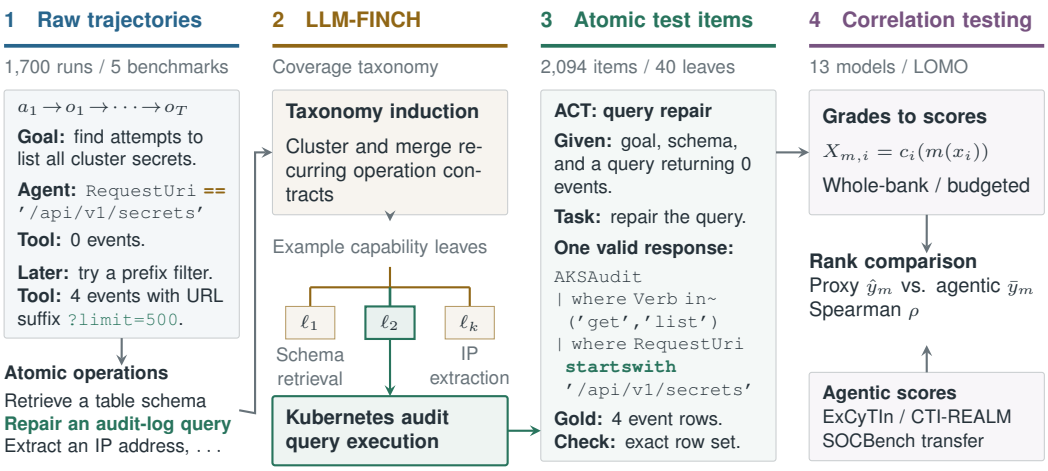


Figure 2: ATOMS TO AGENTS PIPELINE. (1) Recorded agent trajectories are decomposed into atomic operations, such as schema retrieval and query repair. (2) LLM-FINCH clusters and merges operation contracts into a capability taxonomy. (3) Taxonomy leaves guide the construction of self-contained, single-turn items with deterministic checkers. The highlighted Kubernetes example turns a failed query into an ACT item whose repair must recover four recorded events. (4) Atomic-task grades yield proxy scores, evaluated against agentic model rankings under leave-one-model-out evaluation, with SOCBench testing transfer. Trajectory excerpts and leaf labels are abbreviated.

An atomic-task item $i = (x_i, g_i, c_i, \ell_i)$ is a single-turn, deterministically verified evaluation of one operation. The prompt x_i provides the instruction and context. The gold g_i is a reference answer or verification target recovered independently from benchmark answers, recorded environment responses, or executable fixtures. The checker $c_i(\cdot) \in \{0, 1\}$ grades the model response $\hat{y} = m(x_i)$ against g_i, and $\ell_i = \lambda(\omega)$ assigns the item to the operation's taxonomy leaf. For example, to test a SQL query counting failed logins, x_i provides the schema, records, and instruction; g_i is the independently computed count; and c_i executes the query and checks its result. The atomic-task bank $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$ is a fixed collection of self-contained items, partitioned into disjoint leaf-specific sets $\mathcal{I}_\ell = \{i \in \mathcal{I} : \ell_i = \ell\}$ for occupied leaves $\ell \in \mathcal{L}$.

For a source bank S with items indexed by $j \in \{1, \ldots, |S|\}$, let $X_S[m, j] \in [0, 1]$ be model m's normalized score on item i_j. These form the source score matrix $X_S \in [0, 1]^{|\mathcal{M}| \times |S|}$. For our atomic bank, $S = \mathcal{I}$ and $X_\mathcal{I}[m, j] = c_{i_j}(m(x_{i_j})) \in \{0, 1\}$, giving $X_\mathcal{I} \in \{0, 1\}^{|\mathcal{M}| \times |\mathcal{I}|}$. We write $X_{m,i}$ when the source bank and item identity are clear.
Let $\mathcal{M}_{\text{train}}$ and $\mathcal{M}_{\text{eval}}$ denote disjoint calibration and evaluation sets that partition $\mathcal{M}$. Given a source bank S, its score matrix X_S, and an item budget C, our goal is to select $P \subseteq S$ with $|P| = C$ and construct a predictor $\hat{y}_m = f(X_S[m, P])$ using calibration models. For held-out models $m \in \mathcal{M}_{\text{eval}}$, the predictor should approximate the mean agentic score $\bar{y}_m$ while preserving model rankings and pairwise orderings $\mathbf{1}[\bar{y}_m > \bar{y}_{m'}]$. We assess its agreement with $\bar{y}_m$ through Spearman rank correlation ρ over held-out predictions and its ability to recover pairwise orderings $\mathbf{1}[\bar{y}_m > \bar{y}_{m'}]$. Section 2.3 details the objectives and evaluation protocols.

2.2 From Trajectories to Atomic Tasks

We construct the atomic-task bank in two stages (Figure 2). First, we decompose agent trajectories into locally testable operations and organize them into a capability taxonomy, identifying recurring capabilities and structuring their evaluation. Second, we curate self-contained items with independently verified reference answers, allowing these capabilities to be assessed from fixed trajectory states without full agent rollouts.

2.2.1 Decomposition and Taxonomy Construction

To isolate locally testable capabilities within agent workflows, an LLM decomposer maps each trajectory τ to a set of atomic operations $\Omega(\tau) = \{\omega_1, \ldots, \omega_K\}$. The source corpus contains 1,700 runs: 1,428 ExCyTIn threat-investigation runs (Wu et al., 2026), 49 CTI-REALM detection-engineering runs recorded at tool-call resolution (Chakraborty et al., 2026), 144 Cybench CTF runs (Zhang et al., 2025a), 59 CRSBench remediation runs (Chin et al., 2026) and 20 reproduced CyberGym PoC runs (Wang et al., 2026). No SOCBench trajectory contributes an item to the prediction bank. SOCBench operations are inspected only in the post-hoc coverage analysis (Appendix C.3) and scope analysis.

Decomposition. Given a trajectory τ, an LLM decomposer maps it to a set of atomic operations $\Omega(\tau) = \{\omega_1, \ldots, \omega_K\}$. We use *atomic* operationally: each $\omega \in \Omega(\tau)$ specifies a local subgoal, supplied input state, action, resulting output state, and an independently checkable response, but need not be irreducible. Each operation is grounded in supporting trajectory steps E_ω and represented by the fields `name`, `subgoal`, `input_state`, `action`, `output_state`, `verification`, and `trajectory_evidence`, with the final field encoding E_ω. To abstract away domain-specific surface forms, we map each ω to a *canonical operation contract* describing its input–operation–output semantics. Thus, operations such as "extract IOCs from a report" and "parse Metasploit output" can share a parse-and-extract contract, enabling cross-domain unification by function rather than lexical similarity. Any expected value from the decomposer remains an unverified hypothesis; gold targets are established independently during item construction.

Taxonomy construction. We organize the decomposed operations into a hierarchical coverage taxonomy using LLM-FINCH, our extension of the FINCH clustering algorithm (Sarfraz et al., 2019) for automatic discovery of atomic-operation structure. Starting from operation-contract embeddings, LLM-FINCH retrieves candidate neighbors, uses an LLM to assess functional equivalence, and forms clusters through confidence-gated links. Cluster summaries are then recursively grouped into a hierarchy, yielding the leaf set $\mathcal{L}$ and assignment $\lambda(\omega) \in \mathcal{L}$ for each operation ω. The full induction, merge and framework anchoring are described in Appendices A.4 and B.4. The taxon-

omy turns decomposition into an actionable evaluation structure by organizing atomic operations into meaningful cybersecurity capabilities, rather than leaving them as a flat inventory of subtasks (Figure 7). It provides a *capability-normalized ranking signal*, preventing frequently observed operations from dominating rarer capabilities, while simultaneously exposing an *operation footprint* that reveals where the bank is dense, sparse, or blind. This structure therefore serves both evaluation and benchmark design: it supports model comparisons that reflect capability breadth, while identifying concrete coverage gaps where new evaluation items are most informative. We demonstrate this utility on SOCBench, where extending the bank toward taxonomy-identified gaps substantially strengthens its alignment with end-to-end agent performance.

2.2.2 Atomic Task Curation

We turn operations into self-contained evaluation items with fixed input states and independently verified answers. Construction follows four steps for each occupied leaf $\ell \in \mathcal{L}$ and item stream.

(i) Anchoring. Each item is anchored to an operation ω or a supporting step $j \in E_\omega$, inheriting its leaf assignment $\ell_i = \lambda(\omega)$. Item quotas reflect the number of distinct operation names in each leaf, subject to minimum and maximum limits. We exclude anchors lacking prior context or containing temporally inconsistent evidence, and distribute items across early, middle, and late trajectory positions.
(ii) Context and reference answers. The prompt x_i contains only information available before the anchor and excludes private reasoning to prevent answer leakage. Reference answers g_i come from benchmark answers, recorded feedback, validated next actions, or verified absence of evidence. Recorded actions are accepted only when the supplied state rules out alternatives; decomposition-proposed answers require verbatim confirmation in recorded output. Appendix Table 6 summarizes answer provenance.
(iii) Answer construction and checking. Each item receives a checker c_i suited to its question form. Selection items use distractors from the same trajectory and rotate the correct-answer position. Generation items are checked by executing the proposed query or command and comparing its result or effect with the reference target.
(iv) Refinement and admission. Deterministic guards reject invalid candidates. We balance accepted items across leaves, trajectory depths, and outcome classes, then deduplicate by both prompt and item identifier to remove repeated evaluations.

Question-form design. An agent can interpret a tool response correctly yet still choose the wrong next action. This illustrates the abstraction gap identified in the introduction: possessing a capability does not ensure its effective application within a workflow. We therefore design five question forms that test how models apply their capabilities to recognize evidence, select actions, anticipate consequences, generate valid actions, and abstain when evidence is insufficient (Yao et al., 2023; Liu et al., 2024; Yang et al., 2023; Ma et al., 2024). By holding the operation contract fixed while varying the demand placed on the model, these forms distinguish aspects of local competence that end-task outcomes obscure and observation-only evaluation leaves untested (Appendix Table 5).
Recovery items test action selection after a recorded empty result or error. Action-generation items require executable queries or commands whose effects are verified directly (Yang et al., 2023; Liu et al., 2024). Answerability controls withhold supporting evidence and require explicit abstention, denoted $\perp$ and expressed as `INSUFFICIENT` or, for query authoring, `IMPOSSIBLE`; hedged guesses are scored as incorrect (Kirichenko et al., 2026). Together, these forms connect inexpensive static evaluation to the concrete operational demands of agentic workflows.

Bank composition. The prediction bank $\mathcal{I}_{\text{cyber}}$ comprises 2,094 items evaluated across all 13 models: 587 derived from ExCyTIn, 351 from CTI-REALM, and 1,156 from a mixed corpus comprising Cybench, CRSBench, fixtures, and additional instantiated streams. The items span 40 taxonomy leaves and use 12 checker types. This bank supports the primary proxy comparisons and analyses of item composition and answer provenance. Appendix Table 5 reports the question-form distribution and performance across the model panel; Appendix Figure 6 summarizes item sources and counts by form and taxonomy leaf. Overall, 82.9% of items assess action selection, outcome prediction, action generation, or abstention. Action-selection and action-generation items have the highest proportions of discriminating items, at 70.3% and 69.8%, respectively.

2.3 CORRELATION TESTING

Using the notation of Section2.1, we compare the atomic bank $S = \mathcal{I}$ with generic and cybersecurity source banks to test how well their grades predict target scores $\bar{y}_m$. For $P \subseteq S$ with $|P| = C$, the score and pairwise predictors are

$$\hat{y}_m = f(X_S[m, P]) \approx \bar{y}_m, \qquad \hat{b}_{m,m'} = \mathbf{1}[h(X_S[m, P] - X_S[m', P]) > 0] \approx \mathbf{1}[\bar{y}_m > \bar{y}_{m'}]. \tag{1}$$

where f predicts a target score and h a signed preference between models. Bank comparisons assess predictive signal, not a causal effect of trajectory provenance.

Held-out evaluation. In leave-one-model-out (LOMO) evaluation, $\mathcal{M}_{\text{eval}} = \{m^\star\}$ and $\mathcal{M}_{\text{train}} = \mathcal{M} \setminus \{m^\star\}$. Selection and fitting use calibration models only. We pool held-out predictions across folds and report Spearman correlation with $\bar{y}_m$, mean absolute error and pairwise ordering accuracy. Rank correlation is primary; pairwise accuracy excludes tied target scores.

Whole-bank aggregation. Without selection or target calibration, the atomic bank yields the leaf-balanced score: $\hat{y}_m^{\text{pool}} = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \frac{1}{|\mathcal{I}_\ell|} \sum_{j : i_j \in \mathcal{I}_\ell} X_{\mathcal{I}}[m, j]$, which weights occupied capability leaves equally.

Target-blind selection. On $\mathcal{M}_{\text{train}}$, we rank items by their Spearman association with each model's flat mean over items with nonzero calibration variance, without using Y. The selected C items give a mean score standardized against the calibration models. This selection axis differs from the leaf-balanced whole-bank score.

Target-aware selection (PACE). PACE combines local item correlations with calibration target scores and global SVD leverage weighted by those correlations, then fits ridge regression on SVD coordinates (Song et al., 2026). *Frozen* PACE uses a pre-specified configuration; *nested* PACE tunes within $\mathcal{M}_{\text{train}}$ through inner LOMO at $C = 100$. *Auto* PACE tunes using all outer-fold outcomes and reports the median over $C \in \{25, 50, \ldots, 200\}$; it is an exploratory, optimistically selected comparison rather than a fully held-out estimate.

Statistical tests. Whole-bank correlations use target-label permutation tests; target-blind correlations use one-sided asymptotic Spearman tests. For each PACE variant, 100 target-label derangements rerun its selection, fitting and tuning procedure. Auto PACE's matched null does not remove its outer-fold selection optimism. Appendix A.5 details the metrics and tests.

3 EXPERIMENTS

We first evaluate how well atomic-task proxies recover model rankings on agentic benchmarks and quantify their token savings at matched ranking quality (Section 3.1). The two targets represented in bank construction, which we therefore call in-domain, are ExCyTIn, with 589 questions over eight threat-investigation incidents (Wu et al., 2026), and CTI-REALM, with 25 detection-engineering tasks (Chakraborty et al., 2026). These instance counts are distinct from the trajectory-run counts reported in Section2.2. We then compare the atomic bank with PACE's 10-benchmark generic candidate pool (Song et al., 2026) and DefenderBench (Zhang et al., 2025b), and use ablations to assess question-form diversity and operation decomposition (Section3.2). Finally, we test robustness to the choice of selector, evaluate transfer to SOCBench without construction overlap, and examine a taxonomy-guided coverage extension before identifying the proxy's scope limits (Section 3.3). The common model panel comprises Claude-Opus-4.6, Claude-Opus-4.7, Claude-Opus-4.8, DeepSeek-V3.2, GPT-4.1, GPT-4o, GPT-5, GPT-5.2, GPT-OSS-120B, Kimi-K2.5, Nemotron-3-Nano, o3 and o4-mini. Inference configuration is part of model identity. We compare the 2,094-item cyber bank (40 leaves), a 21,464-item generic pool (10 benchmarks), and DefenderBench's 2,499 items (five leaves). These counts and all 806 score files are fixed in a content-hashed manifest; Appendix A.2 specifies the item-identity rule.

Evaluation metrics. Our primary metric is Spearman rank correlation ρ between proxy scores and mean agentic scores across models. We additionally report mean absolute error for absolute-score prediction, pairwise ordering accuracy for model comparisons, and Pearson correlation r in the whole-bank scatter analysis. Evaluation efficiency is measured by marginal token cost at matched ranking quality. Appendix A.5 defines the metrics and statistical tests.

3.1 PREDICTIVE POWER AND COST SAVINGS

Setup. Each model receives one proxy score per candidate pool and one mean agentic score per target, the latter averaged over the target's 589 ExCyTIn questions or 25 CTI-REALM tasks. The budget C is the number of items an estimator may evaluate; $C = 100$ is fixed in advance for the primary comparison, while $C = 25$ is reported separately as the best observed point of a budget sweep. Whole-bank scores require no fitting and are computed for all 13 models directly. The selection-based estimators are evaluated leave-one-model-out: for each of the 13 models, item selection and, for PACE, the fitted predictor use only the other 12 models' grades, and the reported correlation is taken over the 13 held-out predictions.

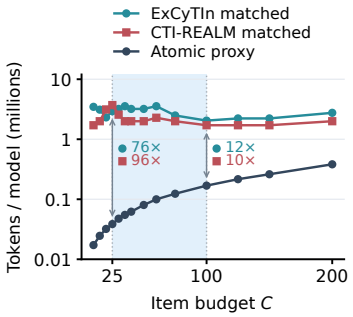
Predictive power. We test whether performance on atomic tasks predicts models' actual agentic benchmark scores, primarily by recovering their rankings on ExCyTIn and CTI-REALM. Following Section2.3, we evaluate three settings: (i) *whole-bank aggregation*, which tests the predictive signal of the complete bank without item selection or target-score calibration; (ii) *target-blind selection*, which tests whether this signal is retained under a limited item budget using only calibration models' bank grades; and (iii) *target-aware selection*, which uses PACE to select items and fit a predictor when calibration models' agentic scores are available. For PACE, we report frozen and nested-tuned configurations alongside exploratory auto tuning. Table 1 compares these settings across the atomic-task, generic, and DefenderBench candidate pools; target scores for held-out models are reserved for evaluation in the frozen and nested settings.

Table 1: Proxy rank recovery on 13 models: Spearman ρ (one-sided p).

Candidate pool	ExCyTIn	CTI-REALM	Mean
Whole-bank (no target scores; all items)			
Atomic-task bank	**0.852** ($p = 0.0003$)	**0.918** ($p < 0.0001$)	**0.885**
Generic	0.808 ($p = 0.0013$)	0.659 ($p = 0.0099$)	0.734
DefenderBench	0.549 ($p = 0.0278$)	0.588 ($p = 0.0178$)	0.569
Target-blind (no target scores; $C = 100$)			
Atomic-task bank	**0.885** ($p < 0.0001$)	0.890 ($p < 0.0001$)	**0.887**
Generic	0.709 ($p = 0.0033$)	0.582 ($p = 0.0184$)	0.646
DefenderBench	0.819 ($p = 0.0003$)	**0.918** ($p < 0.0001$)	0.868
PACE auto† (target scores; $C = 25$–200)			
Atomic-task bank	**0.863** ($p = 0.0099$)	**0.890** ($p = 0.0099$)	**0.877**
Generic	0.808 ($p = 0.0594$)	0.764 ($p = 0.0693$)	0.786
DefenderBench	0.792 ($p = 0.0594$)	0.838 ($p = 0.0099$)	0.815
PACE frozen (target scores; $C = 100$)			
Atomic-task bank	**0.802** ($p = 0.0099$)	**0.808** ($p = 0.0099$)	**0.805**
Generic	0.560 ($p = 0.0396$)	0.423 ($p = 0.0792$)	0.492
DefenderBench	0.462 ($p = 0.0495$)	0.714 ($p = 0.0396$)	0.588
PACE nested (target scores; $C = 100$)			
Atomic-task bank	**0.874** ($p = 0.0099$)	**0.874** ($p = 0.0099$)	**0.874**
Generic	0.676 ($p = 0.0198$)	0.429 ($p = 0.0396$)	0.553
DefenderBench	0.593 ($p = 0.0099$)	0.247 ($p = 0.1980$)	0.420

Mean averages the two targets. Bold marks within-estimator maxima; light blue highlights our bank's maxima, not significant pool differences. †: exploratory outer-fold tuning. Nested tuning and null tests: Appendix A.5.

Figure 3: Marginal cost at matched ranking accuracy across 14 budgets.



Same measured-cost method as Figure 1; 13 models. Target subsets match each proxy's ρ. Shading: $C = 25$–100. Labels: target/proxy token ratios; markers identify benchmarks.

Whole-bank atomic scores achieve Spearman correlations of $\rho = 0.852/0.918$ on ExCyTIn/CTI-REALM. Target-blind selection retains strong predictive power at $C = 100$ ($\rho = 0.885/0.890$; pairwise accuracy $85.9\%/84.6\%$), while the best observed budget, $C = 25$, reaches $\rho = 0.912/0.956$ and $88.5\%/92.3\%$ pairwise accuracy. The whole-bank and fixed-budget target-blind settings attain two-target mean correlations of 0.885 and 0.887, compared with 0.805–0.877 for target-aware PACE on the atomic bank. These differences are descriptive rather than statistically separated, and all three PACE configurations exceed their matched full-pipeline nulls on both targets ($p = 0.01$). Thus, atomic-task grades provide a strong in-domain ranking signal both with and without target-score calibration. In particular, target-blind selection enables screening new models without first obtaining their target-benchmark scores.

Cost savings. We compare measured tokens per model at matched rank correlation on the same 13-model panel. For each proxy we take $N^\star$, the smallest uniformly sampled target subset whose mean correlation with the full benchmark reaches the proxy's, estimated over 10,000 draws, and compare the cost of evaluating that subset with the cost of the proxy. Both sides are measured from the runs underlying the fixed-panel scores and include solver and judge calls, cached input and reasoning tokens. At $C = 25$, the proxy uses 38.6K tokens versus 2.93M for 33 ExCyTIn

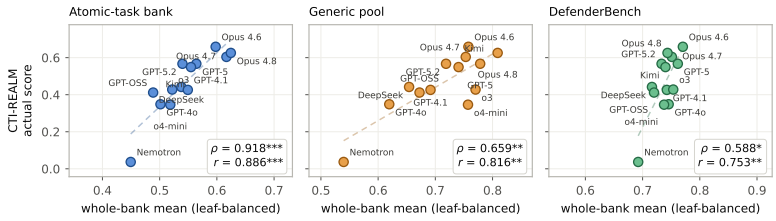


Figure 4: Target-free whole-bank means versus CTI-REALM scores (13 models), with equal horizontal-axis spans across pools. Spearman ρ and Pearson r use one-sided permutation tests: $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$. Test details: Appendix A.5.

questions and 3.71M for 13 CTI-REALM tasks: $76\times$/$96\times$ savings. This outcome-informed budget is exploratory; item selection remains target-blind. The prespecified $C = 100$ uses 168.7K tokens, retaining $12.1\times$/$10.1\times$ savings. Figure 3 covers 14 budgets ($C = 10$–200).

The 80M-token setup cost amortizes after two additional models when replacing both full-target evaluations (59.5M tokens per model) with the $C = 25$ proxy. Including target-validation costs raises the ledger to 854M tokens and payback to 15 models. This full-target comparison is distinct from matched-correlation savings; accounting details appear in Appendix C.4. *To this end, atomic-task proxies recover in-domain model rankings without target-score calibration, with* 76–96$\times$ *lower marginal token cost at the best observed budget.*

3.2 Comparison with other proxies and ablations

Under identical estimators and folds, the atomic bank achieves the highest mean Spearman correlation across the two targets in every evaluation block (Table 1), including 0.885 for whole-bank and 0.887 for target-blind evaluation. DefenderBench performs better on CTI-REALM under target-blind evaluation. These gains remain statistically inconclusive and may partly reflect differences in bank size and composition. Figure 4 provides a model-level view, showing how each uncalibrated whole-bank ranking aligns with the full CTI-REALM benchmark ranking.

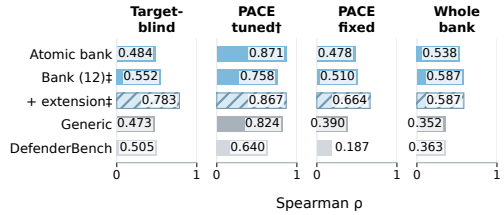
To isolate question-form diversity from pool size, we compare the 359 observation-only items with the full bank and 20 random 359-item mixed-form subsets under target-blind $C = 100$ selection. All conditions share grades, models and targets; target scores are used only for evaluation.

Table 2: Question-form ablation with target-blind selection ($C = 100$).

Condition	Candidate items	ExCyTIn	CTI-REALM
Observation interpretation only	359	0.769 (78.2)	0.703 (76.9)
Full five-form bank	2,094	0.885 (85.9)	0.890 (84.6)
Matched mixed-form subset, median	359	0.874 (84.6)	0.912 (88.5)
Matched-subset wins (ρ and pair)	20 × 359	19/20 (both)	19/20 (both)

Entries are Spearman ρ (pairwise ordering accuracy, %). The matched row is the median across 20 fixed resamples; wins count resamples exceeding the observation-only condition.

Figure 5: SOCBench generalization across candidate pools.



†: exploratory tuning. ‡: 12 models; others 13. Full values and protocols: Appendix A.5.

The full bank and median mixed-form subset outperform observation-only items on both targets; 19 of 20 subsets improve both correlation and pairwise accuracy on each target. This supports form diversity beyond pool size, without establishing a benefit from every form. In matched construction controls, decomposition-curated items reach $\rho = 0.88/0.37$ on CTI-REALM/SOCBench versus $0.73/0.15$ for raw steps, winning 99%/98% of 300 paired draws (cf. Table 2). *In conclusion, our pipeline combines operation decomposition with diverse question forms to assess capabilities within workflows, yielding stronger proxy performance than generic benchmark banks, DefenderBench, and questions curated through trajectory anchoring alone.*

3.3 Robustness and generalization

Ranking robustness beyond PACE. The atomic bank retains its ranking advantage when PACE is replaced by Ridge (Hoerl & Kennard, 1970). Under 13-model LOMO, just 50 atomic items achieve

a two-target mean $\rho = 0.926$, exceeding the best generic (0.860) and DefenderBench (0.805) results across all tested budgets from 50 to 200 items. Across these four budgets, our bank achieves the highest or tied mean correlation under both PACE and Ridge. These results come from a bank of 2,094 candidate items, compared with 21,464 in the generic pool, supporting a compact source of informative proxy tasks. Appendix Table 8 reports the complete comparison, including Lasso and MAE results. Configuration tuning reuses outer-fold outcomes, so the comparisons remain exploratory.

Generalization to a new workflow. Having established in-domain predictive validity and examined ranking robustness, we ask whether the fixed bank transfers to a workflow absent from its construction data. We next evaluate transfer to SOCBench, which contains 240 unit–persona instances and contributes no content to the original bank (DeepTempo, 2026).

Without adaptation, the original bank provides only moderate prediction of SOCBench performance and does not consistently outperform alternative pools. Whole-bank aggregation reaches $\rho = 0.538$ ($p = 0.03$), and target-blind selection at $C = 100$ reaches $\rho = 0.484$ ($p = 0.047$; Figure 5). Although the atomic bank achieves the highest median correlation across 375 prespecified PACE configurations at $C = 100$ (0.478, versus 0.390 for the generic pool and 0.187 for DefenderBench), a separate frozen configuration favors the generic pool (0.698 versus 0.478). Moreover, the high correlation obtained through exploratory target-validated tuning (0.871, $p = 0.01$) falls to 0.456 ($p = 0.06$) under nested tuning. These results indicate that the original bank's predictive signal does not transfer reliably across the tested settings.

Our pipeline's taxonomy provides a concrete path to improvement by identifying missing workflow capabilities. Only 20% of 576 mapped SOCBench operations fall on occupied bank leaves, while 231 map to an unoccupied traffic-profiling task family. Guided by these gaps, we add 219 SQL aggregation items derived from ExCyTIn tables and 96 items addressing residual gaps, expanding the bank from 2,094 to 2,409 items. Comparing both banks on the same 12 models with complete grades for all additions, target-blind correlation at $C = 100$ rises from 0.552 to 0.783, while the median over fixed PACE configurations rises from 0.510 to 0.664 (Figure 5). The taxonomy therefore helps turn limited transfer into actionable guidance for bank expansion. These gains remain exploratory: SOCBench guided the extension and served as its evaluation target, and the displayed comparison lacks matched random additions. Overall, the SOCBench study suggests that cross-workflow generalization depends in part on capability coverage: transfer weakens when the target workflow occupies regions absent from the bank, while taxonomy-guided additions can systematically address those gaps. At the same time, atomic single-turn evaluation does not capture persistence, exploration, or within-incident dependencies, so these results should not be interpreted as predicting every aspect of end-to-end agent behavior.

4 Conclusion

Workflow-derived atomic items provide a strong target-free ranking signal on two in-domain cybersecurity benchmarks. At the fixed target-blind $C = 100$ budget, the atomic-task bank has the highest observed two-target mean correlation (0.887, versus 0.646 for PACE's generic pool and 0.868 for DefenderBench), although the between-pool differences are not statistically established. Beyond aggregate ranking, atomic items with independently verified gold localize model weaknesses to concrete cybersecurity operations. The framework's taxonomy then provides both a leaf-balanced ranking view and a coverage view that exposes missing capabilities and guides targeted bank expansion. A target-blind selector matches nested-tuned PACE in-domain, and target-blind correlations remain strong under the eight tested family holdouts ($0.913/0.949$); frozen PACE weakens to $0.632/0.527$ (Appendix C.1). The focused ablation shows that matched-size mixed-form banks outperform observation-only banks on those targets. Transfer without construction overlap is much weaker, and ranking closely related checkpoints requires separate validation. Atomic tasks are therefore best viewed as an efficient screening instrument for candidate models, with taxonomy-guided coverage used to maintain the bank and end-to-end evaluation retained for calibration, transfer and final decisions. The demonstrated $76\times$–$96\times$ token savings are relative to target-benchmark subsampling, not to the other proxy pools. Future work should directly compare cross-pool efficiency by applying the same budget sweep to all three pools and reporting both correlation at each matched budget and the minimum number of items and tokens required to reach a fixed rank correlation.

AI USE STATEMENT

Large language models were used as components of the method under study (trajectory decomposition, the LLM-FINCH clustering judges, capability and provenance audits) and are described as such in Section 2; every gold label in the bank is derived by execution or from recorded environment output, never from an LLM's assertion. LLM assistance was used for drafting and editing this manuscript and for writing analysis scripts; all reported numbers were produced by checked-in scripts and verified by the authors.

ETHICS STATEMENT

All trajectories were produced by models operating inside the benchmarks' own sandboxed environments on synthetic or previously published incident data; no live systems were targeted. The bank contains security-relevant prompts (log queries, CTF commands, sanitizer reports) of the kind already public in the source benchmarks; it does not contain novel exploit material. One frontier model refused 65% of the bank on a cyber-safety filter, which we report as a measurement issue rather than a model failure.

REPRODUCIBILITY STATEMENT

The item bank rebuilds deterministically from frozen trajectory, taxonomy and cache artifacts with no model calls. The final 13-model view records SHA-256 hashes for 806 files, the panel, pool counts, item-identity rule and code revisions. The taxonomy runs, decompositions, per-model grades, target scores and scripts behind the reported tables and figures accompany the submission.

REFERENCES

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.

Manish Bhatt, Sahana Chennabasappa, Yue Li, Cyrus Nikolaidis, Daniel Song, Shengye Wan, Faizan Ahmad, Cornelius Aschermann, Yaohui Chen, Dhaval Kapil, et al. Cyberseceval 2: A wide-ranging cybersecurity evaluation suite for large language models. *arXiv preprint arXiv:2404.13161*, 2024.

Arjun Chakraborty, Sandra Ho, Adam Cook, and Manuel Meléndez. Cti-realm: Benchmark to evaluate agent performance on security detection rule generation capabilities. *arXiv preprint arXiv:2603.13517*, 2026.

Andrew Chin, Dongkwan Kim, Yu-Fu Fu, Fabian Fleischer, Youngjoon Kim, HyungSeok Han, Cen Zhang, Brian Junekyu Lee, Hanqing Zhao, and Taesoo Kim. Oss-crs: Liberating aixcc cyber reasoning systems for real-world open-source security. *arXiv preprint arXiv:2603.08566*, 2026.

DeepTempo. SOCBench: Evaluating LLM agents as SOC analysts. `https://socbench.org/`, 2026. Accessed: 2026-09-18.

Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 847–864, Philadelphia, PA, August 2024. USENIX Association. ISBN 978-1-939133-44-1. URL `https://www.usenix.org/conference/usenixsecurity24/presentation/deng`.

Luca Gioacchini, Marco Mellia, Idilio Drago, Alexander Delsanto, Giuseppe Siracusano, and Roberto Bifulco. Autopenbench: Benchmarking generative agents for penetration testing, 2024. URL `https://arxiv.org/abs/2410.03225`.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vUaY1t64ZZ.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J Bell. Abstentionbench: Reasoning llms fail on unanswerable questions. *Advances in Neural Information Processing Systems*, 38, 2026.

Hwiwon Lee, Ziqi Zhang, Hanxiao Lu, and Lingming Zhang. Sec-bench: Automated benchmarking of llm agents on real-world software security tasks. *Advances in Neural Information Processing Systems*, 38:116342–116378, 2026.

Guohong Li and Deyi Xiong. Towards optimal evaluation efficiency for large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14176–14183, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.716. URL https://aclanthology.org/2025.emnlp-main.716/.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn LLM agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=4S8agvKjle.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating LLMs with fewer examples. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=qAml3FpfhG.

Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8926–8935. IEEE, 2019.

Minghao Shao, Sofija Jancheska, Meet Udeshi, Brendan Dolan-Gavitt, Haoran Xi, Kimberly Milner, Boyuan Chen, Max Yin, Siddharth Garg, Prashanth Krishnamurthy, et al. Nyu ctf bench: A scalable open-source benchmark dataset for evaluating llms in offensive security. *Advances in Neural Information Processing Systems*, 37:57472–57498, 2024.

Yueqi Song, Lintang Sutawika, Jiarui Liu, Lindia Tjuatja, Jiayi Geng, Yunze Xiao, Daniel Lee, Aditya Bharat Soni, Vincent Lo, Xiang Yue, and Graham Neubig. Pace: A proxy for agentic capability evaluation. *arXiv preprint arXiv:2607.02032*, 2026.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

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.

Shengye Wan, Cyrus Nikolaidis, Daniel Song, David Molnar, James Crnkovich, Jayson Grace, Manish Bhatt, Sahana Chennabasappa, Spencer Whitman, Stephanie Ding, et al. Cyberseceval 3: Advancing the evaluation of cybersecurity risks and capabilities in large language models. *arXiv preprint arXiv:2408.01605*, 2024.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. Cybergym: Evaluating ai agents' real-world cybersecurity capabilities at scale. In *International Conference on Learning Representations*, volume 2026, pp. 123341–123386, 2026.

Yiran Wu, Mauricio Velazco, Andrew Zhao, Manuel Raúl Meléndez Luján, Srisuma Movva, Yogesh K Roy, Quang Nguyen, Roberto Rodriguez, Qingyun Wu, Michael Albada, Julia Kiseleva, and Anand Mudgerikar. ExcyTIn-bench: Evaluating LLM agents on cyber threat investigation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4pQVVSKdkH.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36:23826–23854, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Andy K Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Julian Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny O Oseleononmen, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=tc90LV0yRL.

Chiyu Zhang, Marc-Alexandre Cote, Michael Albada, Anush Sankaran, Jack W Stokes, Tong Wang, Amir Abdi, William Blum, and Muhammad Abdul-Mageed. Defenderbench: A toolkit for evaluating language agents in cybersecurity environments. *arXiv preprint arXiv:2506.00739*, 2025b.

A Setup and construction

A.1 Trajectory corpus

The construction corpus contains 1,700 runs across five benchmarks. Its source-model counts describe trajectory collection, not the 13-model evaluation panel. In particular, the 11-model ExCyTIn source panel is distinct from the final evaluation panel.

Table 3: Trajectory corpus. Every run was executed by an agent against the benchmark's environment; failures are kept.

domain / benchmark	what the agent does	runs	note
Threat investigation: ExCyTIn	SQL over incident telemetry	1,428	70/30 correct/incorrect
Detection engineering: CTI-REALM	11 tools: CTI, ATT&CK, KQL, Sigma	49	tool-call level
CTF: Cybench	Kali terminal	144	human-authored subtask answers
Vuln. reproduction: CyberGym	PoC generation	20	only reproduced runs
Vuln. remediation: CRSBench	reproduce, localise, patch, verify	59	26% of steps fail

A.2 FROZEN DATA MANIFEST

All final-panel analyses use one snapshot containing exactly the 13 models listed in Section 3. The manifest records SHA-256 hashes for 806 CSV files and the revisions of this repository and PACE. It contains 2,094 cyber items in 40 leaves, 21,464 generic items in 10 benchmark leaves and 2,499 DefenderBench items in five leaves. Item identity is $(\text{benchmark}, \text{task}, \text{prompt id})$.

The targets contain 589 ExCyTIn questions, 25 CTI-REALM tasks and 240 SOCBench unit–persona instances. Standard LOMO fits on 12 models and tests on the thirteenth; this training-fold size is not a separate 12-model evaluation. The matched construction control, coverage extension and fine-tune diagnostics explicitly identify their different panels. The headline budget is $C = 25$; the fixed-budget primary comparison uses $C = 100$.

The item-identity rule namespaces repeated generic identifiers by benchmark and task. It retains distinct ACP and LifeBench documents, collapses IFEval metric variants to prompt-level strict accuracy, and keeps the first occurrence of identical GPQA prompts. Applying the rule yields the pool counts reported above.

A.3 ADMISSION GUARDS AND BUILD INVARIANTS

Guards, each added after a real defect: reject if gold appears in the question (*trivial*); in a prior query's text (*leaked*, copyable rather than earned); not in the context (*unanswerable*, inverted for abstention); with fewer than two competing same-type values in the context (*unearned*); alone in the newest result (*read-off*); when the outcome class is not inferable from the prompt (a zero-byte capture, an unstated validator rule); when an arbitrary detail such as a reference `LIMIT` is not stated; when the recorded "action" is agent prose rather than a command. Free-form SQL states the required columns and the row cap so the task is filtering and joining, not guessing a projection. Gold letters cycle across items so "always answer D" scores chance. Depth bands (early / middle / late) are a build parameter with a relaxing quota, because the first version of every stream was made of opening moves; builders are rotated per leaf, because a fixed order once made two streams 71% of the bank; and the bank deduplicates on the rendered prompt as well as the item id, because two anchors can render one prompt and land on two different leaves.

Two dials we expected to tune difficulty (trace length, operand width) were dead against models that know the closed form, and compositional depth gates but does not tune (two chains of identical depth measured $\hat{p} = 1.00$ and 0.00). Per-outcome-class audits catch items that are unanswerable rather than hard: a class scoring at or below chance is a defect signature, and this is how a misparsed error class, an "empty" class that could not be distinguished from an uncaptured output, and a tool-name-only form with several defensible answers were found.

A.4 LLM-FINCH IN FULL

LLM-FINCH constructs a hierarchy of operations by combining embedding-based candidate retrieval with LLM judgments of functional equivalence. The procedure summarised in Section 2.2.1 comprises nine steps:

1. *Embed operation contracts.* Encode each operation's contract text using a configurable embedder and a shared cache.
2. *Retrieve candidate neighbours.* Retrieve the top-k neighbours of each operation, allowing the semantic judge to consider multiple candidates rather than committing to the geometrically nearest one.
3. *Select semantic first neighbours.* An LLM judge selects the candidate that best matches the operation's functional role and assigns a confidence score, replacing FINCH's geometric first-neighbour rule with a semantic decision.
4. *Gate uncertain links.* Retain links that meet the confidence threshold for clustering; log lower-confidence links without merging their endpoints.
5. *Form initial clusters.* Compute connected components using union-find to obtain level-0 clusters. An optional same-contract merge consolidates clusters whose canonical contracts meet a similarity threshold.

6. *Summarise clusters.* An LLM generates a canonical name, description, family, and merged-capability list for each cluster. Summaries follow the operation-record schema, allowing the same procedure to operate at successive levels.

7. *Induce higher levels.* Repeat clustering and summarisation, monitoring cluster counts and operation-pair reassignments for convergence and consulting an LLM stop judge, subject to a maximum depth. The published taxonomy uses the first level whose cluster count falls at or below the configured maximum leaf count.

8. *Attach clusters to parents.* Accept parent attachments only when they meet the attachment-confidence threshold, recording the typed relationship `merge_into_parent`. Otherwise, leave the cluster unattached or mark the proposed relationship as ambiguous.

9. *Assign held-out operations and refine.* Assign operations from a held-out incident post hoc. Low-confidence assignments provide signals for a hierarchy-refinement pass; seed-to-seed reassignment stability is also measured.

Implementation settings. Candidate retrieval uses $k = 20$. Initial links require confidence ≥ 0.55, and optional same-contract merges require similarity ≥ 0.78. Recursive induction is capped at four levels, with an operation-pair reassignment threshold of 0.02 for convergence. Parent attachments require confidence ≥ 0.7; held-out assignments below the same threshold are flagged for refinement.

A.5 Evaluation metrics and protocols

Evaluation metrics. For proxy scores $\hat{y}_m$ and mean agentic scores $\bar{y}_m$, our primary metric is Spearman rank correlation $\rho(\hat{y}, \bar{y})$, which measures agreement between model rankings. Pearson correlation r measures linear association in the whole-bank scatter analysis; for scores on the $[0, 1]$ scale, $\mathrm{MAE}_{\mathrm{pp}} = 100n^{-1}\sum_m |\hat{y}_m - \bar{y}_m|$ measures absolute-score prediction error in percentage points; and pairwise ordering accuracy measures the percentage of evaluated model pairs ordered correctly. Main-text score-based ordering uses concatenated outer-fold predictions, excluding tied target pairs (up to $\binom{13}{2} = 78$ unordered pairs; all 78 are eligible for the reported targets). Appendix C.2 distinguishes this from the separately fitted pairwise objective. We measure efficiency by the target-to-proxy token ratio $R_{\mathrm{tok}}(C) = N^\star(C)k_{\mathrm{inst}}/K_{\mathrm{proxy}}(C)$ at matched ranking quality, where $N^\star(C)$ is the smallest target-subset size whose mean rank correlation over 10,000 uniform draws reaches the proxy correlation (Equation 2). One-time costs are reported in Appendix C.4.

Reading the correlation tables and figure. Table 1 uses target-label permutation tests for whole-bank scores, one-sided asymptotic Spearman tests for the 13-model target-blind scores, and 100 full-pipeline derangements for auto and frozen PACE. Nested PACE uses 100 derangements of the complete nested procedure on its coarse configuration grid. In Figure 4, Spearman ρ uses 100,000 label permutations and Pearson r uses 20,000; both tests are one-sided. Boldface in the correlation tables identifies observed maxima, not statistically established between-pool differences.

SOCBench figure protocols. In Figure 5, *PACE, tuned on target* uses the released cross-fold autotuning rule: configuration choice reuses outer-fold outcomes, and the reported statistic is the median over $C = 25, 50, \ldots, 200$. Its null reruns the pipeline under 100 derangements. *Fixed config* reports the median over 375 pre-specified coarse-grid configurations at $C = 100$, without selecting a configuration by target performance; its null repeats the median statistic on the same 100 derangements. This is not the single frozen configuration in Table 1. *Whole bank* uses leaf-balanced aggregation without selection and a target-label permutation null; *target-blind* uses the fold-local $C = 100$ selector and one-sided asymptotic Spearman tests. The cross-fold-tuned column is exploratory rather than a held-out estimate of configuration choice.

Intervention panel. The intervention in Figure 5 adds 219 taxonomy-guided netflow items (Section 3.3) and 96 gap-fill items for leaves still empty afterwards. The intervention and its matched original-bank reference use 12 models because Claude-Opus-4.6 lacks gap-fill grades; the other candidate-pool rows use 13. On the same 12 models, the original bank gives $0.758/0.510/0.587$ for cross-fold-tuned PACE, the median over fixed PACE configurations and the whole-bank mean, respectively. The extension is included in the leaf-balanced mean as two additional published leaves: $s_m^{\mathrm{ext}} = (40s_m^{\mathrm{bank}} + s_m^{\mathrm{netflow}} + s_m^{\mathrm{gapfill}})/42$. The model scores change by up to 0.0175, but their ordering is unchanged, so Spearman ρ remains 0.587. Target-blind $C = 100$ correlation rises from

0.552 to 0.783 on these same 12 models. Differences between candidate pools are descriptive, not statistically established gains.

Table 4: Full SOCBench correlations and one-sided p-values for Figure 5.

Candidate pool	Target-blind $C = 100$	PACE tuned†	PACE fixed	Whole bank
Atomic-task bank	0.484 ($p = 0.0471$)	**0.871** ($p = 0.0099$)	**0.478** ($p = 0.0198$)	**0.538** ($p = 0.0303$)
Atomic-task bank (12 models)‡	0.552 ($p = 0.0313$)	0.758 ($p = 0.1584$)	0.510 ($p = 0.0297$)	0.587 ($p = 0.0247$)
With intervention‡	0.783 ($p = 0.0013$)	0.867 ($p = 0.0198$)	0.664 ($p = 0.0099$)	0.587 ($p = 0.0247$)
Generic	0.473 ($p = 0.0515$)	0.824 ($p = 0.0396$)	0.390 ($p = 0.0396$)	0.352 ($p = 0.1182$)
DefenderBench	**0.505** ($p = 0.0390$)	0.640 ($p = 0.3168$)	0.187 ($p = 0.1287$)	0.363 ($p = 0.1101$)

†: exploratory tuning. ‡: 12 models; other rows use 13. Bold marks observed maxima among the original pools, not significant differences. Panel matching and protocols are described above.

B Bank and Taxonomy Analysis

B.1 Prediction-Bank Composition

Table 5 details the five question forms, their verification contracts and their empirical profile on the 13-model panel. Figure 6 summarizes the bank's source arms, form distribution and leaf supply, and Appendix B.2 reproduces one item of each form.

Table 5: Five question forms in the 2,094-item bank. Mean accuracy is the item-average pass rate; discriminating items are solved by some but not all 13 models. Both metrics use only bank grades.

Control point	Test → response	Verification contract	Items (%)	Mean acc.	Discrim. (%)
OBSERVE Observation interpretation	supported-state extraction → entity or value	external answer or recorded output; exact/set/context match	359 (17.1)	0.758	154 (42.9)
CHOOSE Action selection	policy choice → candidate action	recorded next action; recovery follows failure; identity match	647 (30.9)	0.436	455 (70.3)
ANTICIPATE Outcome prediction	local outcome → result/empty/error	recorded environment response; parsed-class match	443 (21.2)	0.655	277 (62.5)
ACT Action generation	procedural execution → query or command	independent fixture; execute $\hat{y}$ and compare effects/results	490 (23.4)	0.392	342 (69.8)
ABSTAIN Answerability assessment	support detection → value or $\perp$	mechanical support/verified absence; value or exact-$\perp$ match	155 (7.4)	0.746	78 (50.3)
Total	Five complementary views of an operation contract		**2,094**	**0.550**	**1,306 (62.4)**

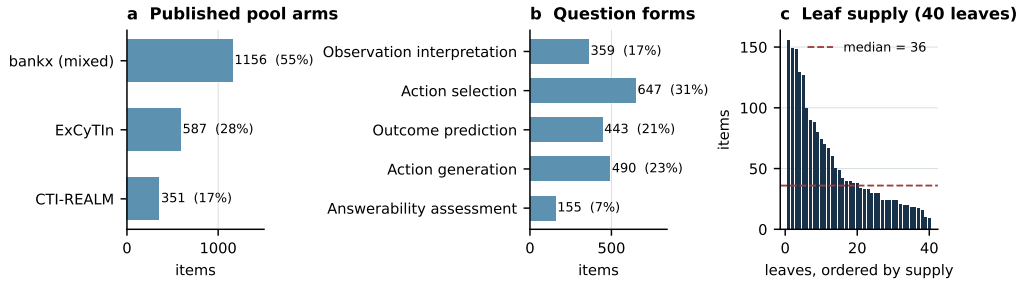


Figure 6: Fixed prediction-bank composition (2,094 items): (a) source arms, (b) five question forms, and (c) item supply across 40 taxonomy leaves. The mixed `bankx` arm includes Cybench, CRS-Bench, fixtures and additional streams.

B.2 EXAMPLE ITEMS FOR EACH QUESTION FORM

Each box below reproduces one item of the fixed 2,094-item bank exactly as the model receives it; the only edits are abridgements marked `[...]`. Gold, checker and source follow each prompt.

Observation interpretation (OBSERVE). The model reads recorded query results and names the entity they implicate; the evidence also holds other candidate entities (an IP address and two process names).

```
You are a SOC analyst closing out an investigation
against a security telemetry database.

# Incident
A security incident where unexpected behavior was observed in a process that ran with no
↪  command line arguments on IP address `202.183.149.174`. This kind of behavior can
↪  indicate code injection or a malicious execution masquerading as a legitimate process.
↪  As part of the investigation into this incident, it was reported that a process
↪  demonstrated suspicious activities such as injecting code into another, leading to
↪  abnormal behaviors like opening a listening port or connecting to a command control
↪  server.

# Everything the investigation established
--- Prior investigation step 1 ---
Query executed:
SELECT * FROM AlertInfo LIMIT 5
Result:
TenantId | TimeGenerated | Timestamp | AlertId | Title | Category | Severity |
↪  ServiceSource | DetectionSource | AttackTechniques | SourceSystem | Type
[... 5 sample AlertInfo rows ...]

--- Prior investigation step 2 ---
Query executed:
SELECT AlertId,Title FROM AlertInfo WHERE Title LIKE '%inject%'
Result:
AlertId | Title
da4bacac3a-d2ba-4b25-89fb-e36b3fd37b1a_1 | Suspicious process injection observed
da240a9e24-7067-4d18-af94-a5ae020c2b30_1 | Suspicious process injection observed

--- Prior investigation step 3 ---
Query executed:
SELECT DISTINCT RemoteUrl,RemoteIP,FileName FROM AlertEvidence WHERE AlertId IN
↪  ('da4bacac3a-d2ba-4b25-89fb-e36b3fd37b1a_1','da240a9e24-7067-4d18-af94-a5ae020c2b30_1')
Result:
RemoteUrl | RemoteIP | FileName
 |  |
https://wcdstaticfilesprdeus.blob.core.windows.net/wcdstaticfiles/WinATP-Intro-Fileless.txt
↪  |  |
 | 72.5.72.208 |
 |  | powershell.exe
 |  | Notepad.exe

# The question this investigation must answer
Related to this alert, can you identify the URL associated with the suspicious process
↪  injection?

# Your answer
You have no further database access. Name the entity the evidence above actually
↪  implicates.

The evidence contains other values of the same kind, including well-known tools and system
binaries. The one that is most recognisable is usually not the one being asked for --
↪  commit to
what the records support, not to what looks familiar.

Answer with a host or file name, and nothing else -- no explanation, no surrounding prose.

End your reply with exactly one line:
ANSWER: <value>
```

Gold: `https://wcdstaticfilesprdeus.blob.core.windows.net/wcdstaticfiles/winatp-intro-fileless.txt`. *Checker:* the parsed answer must equal the gold value (case-insensitive, surrounding quotes and markup ignored). *Source:* ExCyTIn, `ex40_excytin_commit`; item `cmt_2f5b83ed2a`.

Action selection (CHOOSE), recovery variant. After a recorded failure, the model picks the step the investigation actually took next from four real candidates; the three distractors were tried in other runs and led nowhere.

```
You are a SOC analyst continuing an in-progress investigation.

# Incident
A security incident where unexpected behavior was observed in a process that ran with no
↪  command line arguments on IP address `202.183.149.174`. This kind of behavior can
↪  indicate code injection or a malicious execution masquerading as a legitimate process.
↪  As part of the investigation into this incident, it was reported that a process
↪  demonstrated suspicious activities such as injecting code into another, leading to
↪  abnormal behaviors like opening a listening port or connecting to a command control
↪  server.

# Evidence gathered so far
--- Prior investigation step 1 ---
Query executed:
SELECT DeviceName, DeviceId FROM DeviceInfo WHERE PublicIP = '202.183.149.174' LIMIT 10
Result:
DeviceName | DeviceId

--- Prior investigation step 2 ---
Query executed:
SELECT DeviceName, DeviceId FROM DeviceNetworkInfo WHERE IPAddresses LIKE
↪  '%202.183.149.174%' LIMIT 10
Result:
DeviceName | DeviceId

# Your last action FAILED
You just ran:

SELECT * FROM DeviceProcessEvents WHERE ProcessCommandLine = '' OR ProcessCommandLine IS
↪  NULL LIMIT 20

and it returned:

TenantId | AccountDomain | AccountName | AccountObjectId | AccountSid | AccountUpn |
↪  ActionType | AdditionalFields | AppGuardContainerId | DeviceId | DeviceName | FileName
↪  | FolderPath | FileSize | In [...]

# Goal
Related to this alert, can you identify the URL associated with the suspicious process
↪  injection?

# Candidate recovery actions
One of these is what the investigation actually did next to get unstuck. The others were
↪  tried
in real investigations and led nowhere.

A. SELECT DeviceName, ActionType, RemoteIP, RemoteUrl, InitiatingProcessFileName,
↪  InitiatingProcessCommandLine, InitiatingProcessId FROM DeviceNetworkEvents WHERE
↪  RemoteIP = '202.183.149.174' OR LocalIP = '202.183.149.174' LIMIT 20

B. SELECT * FROM DeviceFileEvents WHERE DeviceName =
↪  'vnevado-win11a.vnevado.alpineskihouse.co' AND InitiatingProcessId = 8932 LIMIT 20

C. SELECT * FROM DeviceEvents WHERE DeviceName = 'vnevado-win11a.vnevado.alpineskihouse.co'
↪  AND ActionType LIKE '%Inject%' LIMIT 20

D. SELECT ActionType, FileName, FolderPath, InitiatingProcessFileName,
↪  InitiatingProcessCommandLine, RemoteUrl, RemoteIP, AdditionalFields FROM DeviceEvents
↪  WHERE DeviceName = 'vnevado-win11a.vnevado.alpineskihouse.co' LIMIT 50

Which one recovers from this failure and makes progress?

End your reply with exactly one line:
ANSWER: <letter>
```

Gold: `A`. *Checker:* the parsed answer must equal the gold value (case-insensitive, surrounding quotes and markup ignored). *Source:* ExCyTIn, `ex40_excytin_recovery`; item `rec_96ca1578b5`.

Outcome prediction (ANTICIPATE). Given the trajectory so far and the next tool call, the model predicts the recorded environment response class.

```
You are a detection engineer working toward a detection objective.
```

```
# Objective
Write a detection rule to identify nested shell obfuscation by monitoring for bash or sh
↪  processes with multiple levels of bash -c invocations in the command line, indicating
↪  layered command execution for evasion

# What you have done so far
--- Step 1 ---
Called: list_cti_report_tags({})
Returned: { "available_tags": [ "detection-engineering", "linux", "security",
↪  "persistence", "privilege-escalation", "kubernetes", "cloud", "rbac", "azure",
↪  "auditd", "containers", "threat-detection", [...]

# The call you are about to make
search_sigma_rules({"keyword": "bash -c bash -c", "limit": 10})

# Question
What will this call return?

  results   rows or records come back
  none      the call succeeds but returns nothing
  error     the call fails

Answer with exactly one of: results, none, error
```

Gold: `none`. *Checker:* the parsed answer must equal the gold value (case-insensitive, surrounding quotes and markup ignored). *Source:* CTI-REALM, `ctir4p_cti_realm_leaf_call_outcome`; item `ctlo_e599da6783`.

Action generation (ACT). The model writes the query itself; it is executed and graded on what it returns, not on its phrasing.

```
You are a SOC analyst continuing an in-progress investigation
against a security telemetry database.

# Incident
A recent security incident was flagged due to the execution of a known credential theft
↪  tool, which was used to dump user's credentials. This was detected on an account from a
↪  specific IP address.

# Evidence gathered so far
--- Prior investigation step 1 ---
Query executed:
SELECT AlertId, Title FROM AlertInfo WHERE Title LIKE '%Mimikatz%'
Result:
AlertId | Title
dae2b9ec2a-af6f-461e-aa1e-f33b657f7ccd_1 | Mimikatz credential theft tool
da1b596080-1f29-4f4e-a431-173e1ce073ef_1 | Mimikatz credential theft tool

# What this step must establish
the 5 distinct file name values -- Retrieve file names associated with the identified
↪  Mimikatz alerts

# Your next query
Write one SQL SELECT query that establishes it.

You choose the table, the joins, the filtering and the columns -- there is no expected
↪  phrasing, and
a query that reaches the same finding another way passes. Work out from the evidence above
↪  which
records bear on it.

Your query runs against the real incident database. It passes if what it returns contains
↪  the
value(s) named above AND is scoped to them: at most 12 rows. Two things fail -- a query
↪  that
misses any of them, and one that dumps a table and leaves them to be spotted by eye.

End your reply with the query on its own, in a ```sql fenced block.
```

Gold: `mimidrv.sys`, `mimikatz.exe`, `mimilib.dll`, `mimilove.exe`, `mimispool.dll`. *Checker:* the query is executed on the incident database; it passes if the result contains every gold value and has at most the stated number of rows (unfiltered whole-table reads fail). *Source:* ExCyTIn, `ex40_excytin_freeform_action`; item `ff2_6c6afb9edc`.

Answerability assessment (ABSTAIN). The evidence needed to answer is withheld; the model must say so rather than guess.

```
You are a SOC analyst continuing an in-progress investigation.

# Incident
On 2024-07-22, at 08:45:32.664841+00:00, a sign-in from an anonymous IP address
↪  95.202.65.202 was detected. The account associated with this sign-in has the AadUserId
↪  `89e933b9-5b2e-4bd6-bcdf-033db707ef82`. This incident might indicate potential
↪  malicious activity leveraging the anonymized IP.

# Evidence gathered so far
--- Prior investigation step 1 ---
Query executed:
SELECT AppId FROM CloudAppEvents WHERE AccountObjectId =
↪  '89e933b9-5b2e-4bd6-bcdf-033db707ef82' LIMIT 5
Result:
[tool error] {"type": "unknown", "message": "OperationalError: no such column: AppId"}

--- Prior investigation step 2 ---
Query executed:
PRAGMA table_info(CloudAppEvents)
Result:
columns: TenantId, AccountId, AccountType, AdditionalFields, RawEventData, ReportId,
↪  ObjectId, ObjectType, ObjectName, ActivityObjects, ActivityType, UserAgent, ISP, City,
↪  CountryCode, IsAnonymousProxy, IsExternalUser, IsImpersonated, IPAddress, IPCategory,
↪  IPTags, OSPlatform, DeviceType, IsAdminOperation, AccountDisplayName, AccountObjectId,
↪  AppInstanceId, ApplicationId, Application, ActionType, UserAgentTags, TimeGenerated,
↪  Timestamp, SourceSystem, Type

# Your task
What is the application id? Retrieve records related to suspicious inbox manipulation
↪  actions for the given AccountObjectId

You have NO further database access -- reason only from the evidence above. The evidence
↪  may be
incomplete: it might not contain what is needed to answer. Do NOT guess a plausible-looking
↪  value.

If the evidence above is sufficient, answer with the value.
If it is NOT sufficient, answer with exactly: INSUFFICIENT

End your reply with exactly one line:
ANSWER: <value or INSUFFICIENT>
```

Gold: `INSUFFICIENT`. *Checker:* gold is `INSUFFICIENT`, so the answer must begin with it; answering with a value fails. *Source:* ExCyTIn, `ex40_excytin_abstention`; item `abst2_ff508cd193`.

B.3 Where gold comes from

Gold targets come from recorded benchmark evidence or independent execution, except for 33 planning items whose labels are decomposition-derived subgoals. Table 6 classifies the fixed 2,094-item bank using its released gold-source metadata and inspection of item records and construction code. For the 67 items without a `gold_source` field, this inspection identifies 33 recorded trajectory queries, 33 decomposition-derived subgoals and one executed fixture output; the last is included in the fixture-output category. These source categories differ from question forms: one form can use several sources of gold.

B.4 The induced inter-domain taxonomy

Figure 7 shows the first two induced levels of the merged inter-domain taxonomy of Section 2.2.1: nine emergent groups (Level 1) over 22 capability families (Level 2), with the number of leaf capabilities under each. The 172 leaves themselves, their per-domain provenance and framework anchors are released with the bank. This broader coverage hierarchy is distinct from the 40 leaf directories used to aggregate the fixed 2,094-item prediction bank; the two counts describe different taxonomy views.

Table 6: **Gold-answer provenance for the fixed 2,094-item prediction bank.** Categories summarize the released `gold_source` field, supplemented by inspection of item records and construction code where that field is absent.

Gold source	Items	Example evaluation
Action taken next in a successful trajectory	386	Next-action MCQ; distractors are steps that led nowhere
Execution against the real incident database	304	Free-form query authoring; `values_in_result`
Executed output of a fixture function or query	283	Synthetic code and SQL fixtures
Recorded environment feedback class	394	Tool-outcome prediction: will this query error?
Benchmark-provided answer or subtask answer	186	Investigation answers; Cybench subtasks
Recorded tool call or next command	175	CTI-REALM next call; CRSBench next command
Verified absence of an answer in the evidence	155	Abstention; mixed 50/50 with answerable controls
Consensus of successful runs, verified by execution	82	Commit-own-retrievals; KQL advances the objective
Decomposition-selected answer verified in recorded output	27	Verbatim recorded-output verification
Executed result set	20	Result-set verification
Benchmark-graded episode	16	Episode-level benchmark grade
Recorded trajectory query	33	Execute candidate and reference queries; compare retrieved evidence
Decomposition-derived subgoal	33	Next-subgoal MCQ; label from the trajectory decomposition
Total	**2,094**	**Fixed prediction bank**

B.5 Difficulty calibration

Difficulty is a property of the panel. On the fixed 13-model panel, 1,306 of the 2,094 items (62.4%) have $0 < d_i < 1$; 307 are missed by every model and 481 are solved by every model. The item-average pass rate is 0.550. These are the same descriptive bank statistics reported in Section 2.2.2 and Table 5, not estimates of downstream predictive validity. Binary grades on 13 models produce 14 possible pass-rate levels. An independent post-freeze provenance audit reviewed all 587 ExCyTIn items and recommended dropping 182. Those recommendations were not applied to the fixed bank analyzed here; the complete verdict file is released and this unresolved audit debt is a limitation.

C Additional results

C.1 Selection and robustness experiments

Target-free scores. All floors are recomputed on the final common panel. The cyber leaf-balanced mean gives $\rho = 0.852/0.918/0.538$ on ExCyTIn, CTI-REALM and SOCBench, compared with $0.808/0.659/0.352$ for generic and $0.549/0.588/0.363$ for DefenderBench. The cyber result is significant on both in-domain targets ($p < 0.001$) and marginal on SOCBench ($p = 0.030$). Flat item means are reported separately because large benchmark leaves otherwise dominate the generic score.

Frozen-configuration null. For each of 100 target-label derangements we repeat fold-local item selection, SVD construction, fitting and prediction. With $(q, m, n_h, n_d) = (0.5, 0.5, 2, 7)$ fixed, the cyber bank gives $\rho = 0.802/0.808/0.478$; null 95th percentiles are $0.429/0.374/0.401$ and empirical $p = 0.01/0.01/0.06$. Generic is stronger on SOCBench (0.698, $p = 0.01$), an important counterexample to uniform cyber-bank superiority.

Paired pool differences. We resample the 13 held-out models 5,000 times, applying each draw to both pools. Cyber-minus-generic differences for target-blind selection are 0.176 (95% CI

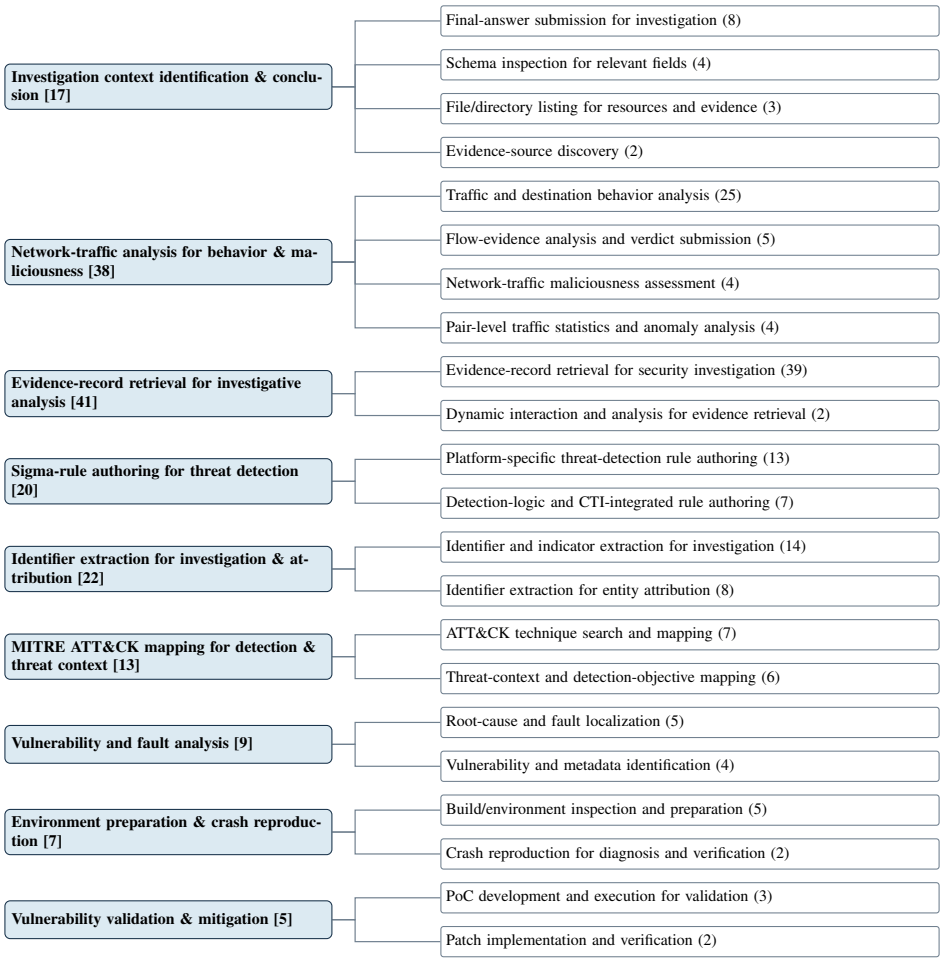


Figure 7: First two levels of the inter-domain taxonomy (labels abbreviated). Brackets and parentheses count leaf capabilities under Level-1 groups and Level-2 families, respectively.

$[-0.142, 0.558]$), 0.308 ($[-0.073, 0.764]$) and 0.011 ($[-0.542, 0.621]$) on ExCyTIn, CTI-REALM and SOCBench. Frozen-PACE differences are 0.242 ($[-0.261, 0.804]$), 0.385 ($[-0.209, 1.038]$) and -0.220 ($[-0.813, 0.342]$). Every interval includes zero, so pool-to-pool superiority is not established on this panel.

Family holdout. Eight groups are fixed before evaluation: Claude Opus, GPT-4, GPT-5, the o-series, DeepSeek, Kimi, GPT-OSS and Nemotron. Under target-blind selection, cyber correlations are $0.913/0.949/0.520$ with jackknife-over-group ranges $[0.890, 0.932]/[0.925, 0.956]/[0.273, 0.706]$. Frozen PACE gives $0.632/0.527/0.544$ and empirical null $p = 0.03/0.03/0.02$. We treat these as robustness estimates over eight groups rather than evidence that family holdout improves performance.

Exploratory cross-fold tuning. The released auto-tuning rule chooses one configuration after scoring all 13 outer predictions. Its medians over $C = 25, 50, \ldots, 200$ are $0.863/0.890/0.871$ for cyber, $0.808/0.764/0.824$ for generic and $0.792/0.838/0.640$ for DefenderBench. A matched 100-derangement null reruns tuning and selection within every draw; cyber clears it on all three targets ($p = 0.01/0.01/0.01$), generic only on SOCBench ($p = 0.06/0.07/0.04$), and DefenderBench only on CTI-REALM ($p = 0.06/0.01/0.32$). These are optimistically selected estimates: the SOCBench cyber value falls to 0.456 when tuning is nested. We report the sweep as a sensitivity analysis and do not use it as held-out evidence.

C.2 LOOCV SELECTION BASELINES

Here LOOCV denotes leave-one-model-out evaluation, as in the main text. MAE and ρ evaluate absolute-score predictions; the pair column evaluates the separately fitted pairwise objective over $13 \times 12 = 156$ ordered comparisons involving a held-out model (each of the 78 unordered pairs appears in both directions), not the ordering of concatenated absolute-score predictions. These 156 comparisons are not independent model pairs. Its percentages therefore need not match the score-based pairwise accuracy in the main tables.

We compare PACE with Lasso (Tibshirani, 1996) and Ridge (Hoerl & Kennard, 1970) on the final 2,094-item bank and fixed 13-model panel at $C \in \{50, 100, 150, 200, 250\}$. In every LOOCV fold, Lasso selects its nonzero coefficients, while Ridge ranks items by coefficient magnitude (optionally multiplied by the item's training-fold standard deviation), retains exactly C items, and refits on that subset. Lasso cannot enforce an exact budget, so Table 7 reports its mean selected count; its repeated rows reflect the same fitted sparsity rather than duplicated budget-constrained runs. Lasso and Ridge configurations are chosen by minimum LOOCV MAE over raw/standardized features and regularization grids; PACE uses its matching-C cross-fold selection rule. Because configuration choice reuses the outer-fold outcomes, all three are optimistically selected controls rather than leak-free estimates.

Table 7: PACE, Lasso and Ridge selection across $C = 50$–250 on the fixed 2,094-item bank (13-model LOOCV). MAE is in percentage points; pairwise accuracy is in percent. Selected items are means across folds. Bold marks the best result within each target–budget block and metric. Lasso uses the fully converged MAE-selected grid; its pairwise predictor is refitted on the corrected selections.

ExCyTIn

	$C = 50$			$C = 100$			$C = 150$			$C = 200$			$C = 250$		
Metric	PACE	Lasso	Ridge	PACE	Lasso	Ridge	PACE	Lasso	Ridge	PACE	Lasso	Ridge	PACE	Lasso	Ridge
Selected	50	41.2	50	100	41.2	100	150	41.2	150	200	41.2	200	250	41.2	250
MAE ↓	8.04	8.73	**7.33**	**7.61**	8.73	8.16	**7.05**	8.73	8.41	**7.25**	8.73	9.06	**7.23**	8.73	9.10
ρ ↑	0.797	0.841	**0.868**	0.835	**0.841**	0.835	**0.868**	0.841	0.863	**0.874**	0.841	0.835	**0.874**	0.841	0.813
Pair ↑	82.05	**86.54**	**86.54**	82.05	86.54	**87.18**	85.90	86.54	**89.74**	85.90	**86.54**	85.90	85.90	**86.54**	**86.54**

CTI-REALM

	$C = 50$			$C = 100$			$C = 150$			$C = 200$			$C = 250$		
Metric	PACE	Lasso	Ridge	PACE	Lasso	Ridge	PACE	Lasso	Ridge	PACE	Lasso	Ridge	PACE	Lasso	Ridge
Selected	50	43.6	50	100	43.6	100	150	43.6	150	200	43.6	200	250	43.6	250
MAE ↓	**6.00**	8.81	7.38	**5.90**	8.81	7.43	**6.00**	8.81	8.26	**6.34**	8.81	8.41	**6.55**	8.81	8.40
ρ ↑	0.907	0.604	**0.923**	**0.918**	0.604	0.852	**0.879**	0.604	0.769	**0.874**	0.604	0.857	**0.918**	0.604	0.786
Pair ↑	87.18	87.82	**91.67**	88.46	87.82	**91.03**	84.62	87.82	**91.03**	84.62	87.82	**92.95**	88.46	87.82	**92.95**

PACE has the lowest MAE in nine of ten target–budget blocks and the highest or tied rank correlation in seven. Ridge is strongest at $C = 50$ and gives the highest or tied pairwise accuracy in nine of ten blocks. Lasso's selected count is controlled by its penalty rather than C, yielding the same sparse solution at each displayed cap. These results support PACE as a competitive joint selector and predictor, but do not establish significance between methods.

Cross-bank robustness. Table 8 compares the frozen atomic, generic and DefenderBench pools at $C \in \{50, 100, 150, 200\}$. Each LOMO fold fits on 12 models and predicts the thirteenth. All 806 score-file hashes match the frozen manifest. PACE retains its archived budget-specific configuration, selected by minimizing $100\,\mathrm{MAE} - \rho$ with MAE on the $[0, 1]$ scale. Ridge selects the configuration maximizing LOMO Spearman correlation over the same raw/standardized regularization grid; an independent SVD-based final fit reproduces every reported Ridge correlation.

For Lasso, we rerun the complete grid of 30 penalties and two feature scalings for each bank–target pair, checking convergence in every fold before selecting a configuration. All 360 configurations (4,680 fold fits) satisfy the dual-gap tolerance of 10^{-6} times the training-target variance. Fits that do not converge at the original iteration limit are continued until this criterion is met. Table 8 maximizes LOMO Spearman correlation to match the ranking objective; Table 7 retains minimum-

MAE selection and refits the pairwise predictor on the corrected selections. Lasso's penalty-selected fit is repeated across budgets; these columns do not constitute additional experiments.

The atomic bank has the highest two-target mean correlation at every budget under PACE and the highest or tied mean under Ridge (tied with generic at $C = 150$). With just 50 selected items, Ridge reaches a mean $\rho = 0.926$, exceeding the best generic (0.860) and DefenderBench (0.805) results over all four tested budgets. Averaging over budgets and targets gives $\rho = 0.869/0.889$ for PACE/Ridge, versus $0.798/0.854$ for generic and $0.795/0.781$ for DefenderBench. Thus, a bank with roughly one-tenth as many candidate items as generic retains stronger average ranking quality under both methods. Under fully retuned Lasso, the means are 0.736 for atomic, 0.772 for generic and 0.734 for DefenderBench, so the advantage is not uniform across estimators. Generic has the lowest overall MAE under all three methods. PACE retains its original objective; both selection and prediction differ between methods. Configuration choice reuses outer-fold outcomes, and the observed differences are not statistically established.

Table 8: Ranking quality across candidate banks on the fixed 13-model panel (LOMO). PACE uses its original composite objective; Ridge and Lasso maximize Spearman correlation over their full grids, with convergence checked for every Lasso fit. Avg. averages four budgets; Overall averages both targets. Bold marks the best bank within each method and column.

(a) Spearman $\rho \uparrow$

		ExCyTIn					CTI-REALM					
Method	Bank	50	100	150	200	Avg.	50	100	150	200	Avg.	Overall
PACE	Atomic bank	0.797	**0.835**	**0.868**	**0.874**	**0.844**	**0.907**	**0.918**	0.879	**0.874**	**0.895**	**0.869**
	Generic	**0.819**	0.769	0.797	0.841	0.806	0.863	0.692	**0.923**	0.681	0.790	0.798
	DefenderBench	0.615	0.797	0.813	0.813	0.759	0.830	0.841	0.835	0.813	0.830	0.795
Ridge	Atomic bank	**0.918**	**0.890**	0.863	0.874	0.886	**0.934**	**0.901**	0.852	**0.885**	**0.893**	**0.889**
	Generic	0.907	0.863	**0.918**	**0.890**	**0.894**	0.813	0.830	0.797	0.819	0.815	0.854
	DefenderBench	0.775	0.747	0.670	0.681	0.718	0.835	0.852	**0.857**	0.830	0.843	0.781
Lasso†	Atomic bank	**0.868**	**0.868**	**0.868**	**0.868**	**0.868**	0.604	0.604	0.604	0.604	0.604	0.736
	Generic	0.857	0.857	0.857	0.857	0.857	0.687	0.687	0.687	0.687	0.687	**0.772**
	DefenderBench	0.775	0.775	0.775	0.775	0.775	**0.692**	**0.692**	**0.692**	**0.692**	**0.692**	0.734

(b) MAE (percentage points) $\downarrow$

		ExCyTIn					CTI-REALM					
Method	Bank	50	100	150	200	Avg.	50	100	150	200	Avg.	Overall
PACE	Atomic bank	8.04	**7.61**	7.05	7.25	7.49	6.00	5.90	6.00	6.34	6.06	6.77
	Generic	**6.38**	**7.61**	**6.93**	**6.35**	**6.82**	**5.00**	**5.55**	**4.49**	**6.15**	**5.30**	**6.06**
	DefenderBench	8.76	8.66	8.43	8.58	8.61	6.46	6.76	6.11	6.39	6.43	7.52
Ridge	Atomic bank	9.30	8.69	8.41	9.50	8.98	**7.56**	9.28	8.68	**8.59**	**8.53**	8.75
	Generic	**7.28**	**8.63**	**8.19**	**8.89**	**8.25**	9.03	9.12	**7.68**	9.00	8.71	**8.48**
	DefenderBench	11.89	12.80	11.68	12.07	12.11	8.01	**8.83**	8.99	9.63	8.87	10.49
Lasso†	Atomic bank	9.71	9.71	9.71	9.71	9.71	9.01	9.01	9.01	9.01	9.01	9.36
	Generic	**7.26**	**7.26**	**7.26**	**7.26**	**7.26**	8.81	8.81	8.81	8.81	8.81	**8.04**
	DefenderBench	11.70	11.70	11.70	11.70	11.70	**8.69**	**8.69**	**8.69**	**8.69**	**8.69**	10.20

Banks contain 2,094 / 21,464 / 2,499 items (atomic / generic / DefenderBench). PACE and Ridge select exactly C items. †Lasso's penalty determines sparsity; the same fit is repeated across budgets. Mean selected counts (ExCyTIn / CTI-REALM): 39.5 / 63.8, 32.5 / 12.5, 13.8 / 36.0, respectively. The complete Lasso sweep comprises 360 configurations and 4,680 converged fold fits across banks and targets. PACE was not retuned for Spearman. Tuning reuses outer-fold outcomes; comparisons remain exploratory, and boldface does not establish significance. Averages are descriptive, not independent replications. PACE retains archived precision.

Diagnosing Lasso's CTI-REALM weakness. The converged, Spearman-selected Lasso reaches $\rho = 0.868$ on ExCyTIn but only 0.604 on CTI-REALM. To examine the prediction step, we retain its selected items and fold-local scaling and refit Ridge, choosing the refit penalty from the existing 25-value grid. CTI-REALM correlation rises to 0.742, showing that the selected items retain ranking information that the Lasso predictor does not recover. This diagnostic supports a contribution from regularization and coefficient weighting; it does not isolate a complete causal explanation. Full Ridge selection at $C = 50$ reaches 0.934 on CTI-REALM, but changes both selection and budget. On ExCyTIn, the same refit reduces correlation to 0.621, so refitting is not uniformly beneficial. These additional comparisons also reuse outer-fold outcomes for tuning.

C.3 TAXONOMY-GUIDED COVERAGE INTERVENTION

The 10-model netflow-only diagnostic below predates the final-panel comparison. Its $\rho = 0.31$ baseline is not the $\rho = 0.484$ target-blind result on the final 13-model panel. Figure 5 separately reports the bank plus 219 netflow and 96 gap-fill items on the 12 models with complete extension grades; the original bank remains fixed at 2,094 items.

The main results use the taxonomy's ranking signal through leaf-balanced aggregation. This appendix examines its second role: coverage signal. SOCBench's taxonomy footprint was inspected before the intervention, and diagnosis and evaluation both use SOCBench. The estimator comparison reported here does not include a matched random-additions arm. The results therefore show that the taxonomy can identify a useful construction backlog, but do not isolate the predictive advantage of taxonomy-guided additions against equally sized random additions.

Taxonomy footprint and selection. For target benchmark B', define its operation footprint $\phi_{B'}(\ell)$ as the fraction of decomposed target operations assigned to leaf ℓ, and coverage $\kappa(B') = \sum_{\ell:|\mathcal{I}_\ell|>0} \phi_{B'}(\ell)$. This coverage statistic measures represented operation mass, not predictive validity. The primary target-blind selector uses training-fold item discrimination, not footprint-based quotas; the footprint is used here to diagnose missing item supply.

Coverage diagnosis and intervention. SOCBench contributes no items to the original bank. Coverage is computed from one frozen assignment–crosswalk pair containing 576 mapped operations from its gold trajectories. Initial coverage is $\kappa(\text{SOCBench}) = 0.20$; 231 operations map to *Traffic Profiling for Host and Pair Analysis*, an unoccupied leaf in the construction view used for this diagnostic. The earlier 10-model comparison reports a best target-blind baseline of $\rho = 0.31$ on its pre-intervention supply, not on the final 13-model evaluation. We use this diagnosis to construct 219 SQL aggregation items covering eight operation forms (including host roll-up, destination ranking, port patterns and process attribution). The items execute against ExCyTIn network tables, and reference queries are admitted only when their results are unique with margin; no SOCBench observation or answer is included in an item. Table 9 reports the resulting unselected pool scores.

Table 9: SOCBench after construction. Netflow pool mean, no selection, 10 models; "before" is the best target-blind selection on the old supply.

	wide	detection eng.	adversary hunter	SOC analyst	threat analyst
netflow pool mean	$+\mathbf{0.69}$ ($p{\approx}0.01$)	$+0.80$	$+0.60$	$+0.59$	$+0.26$
best blind before	$+0.31$	$+0.62$	$+0.32$	$+0.21$	$+0.12$

The added pool reaches $\rho = 0.69$ on aggregate SOCBench and 0.80, 0.60, 0.59 and 0.26 on its four personas, compared with 0.31, 0.62, 0.32, 0.21 and 0.12 before construction. Its correlation with ExCyTIn is 0.18. Re-merging raises measured SOCBench coverage from 0.20 to 0.76. Together, the footprint change and the added pool's model correlation show the two signals operating separately: the taxonomy detects a missing capability region, while the instantiated items measure model variation within it. Because the intervention was prompted and evaluated by SOCBench, these values are post-hoc diagnostics and do not enter the primary predictive-validity claim.

C.4 COST BREAKDOWN AND PAYBACK

We quantify evaluation efficiency as the marginal token cost per model at *matched rank correlation*, using the same 13-model panel for cost and accuracy measurements. Let $K_{\text{proxy}}(C)$ denote the mean cost of evaluating C selected atomic items and k_{inst} the mean cost per target-benchmark instance. Both are derived from recorded usage, including solver and judge calls, cached input and reasoning tokens. For each target, $\overline{\rho}_{\text{target}}(N)$ is the mean Spearman correlation between rankings from N uniformly sampled instances and the full benchmark, estimated over 10,000 draws. We define the minimum matching subset size and the corresponding target-to-proxy cost ratio as

$$N^\star(C) = \min\left\{N : \overline{\rho}_{\text{target}}(N) \geq \rho_{\text{bank}}(C)\right\}, \qquad R_{\text{tok}}(C) = \frac{N^\star(C) k_{\text{inst}}}{K_{\text{proxy}}(C)}. \tag{2}$$

Headline operating point and measurement boundary. Figure 1 shows $C = 25$, the best observed point in the target-blind budget sweep, rather than a budget fixed before target evaluation. Each held-out model's items are selected using only the other models' bank grades. On the same 13-model panel, 10,000 uniform target-subsample draws establish matched-quality requirements of 33 ExCyTIn questions and 13 CTI-REALM tasks. The proxy averages 38.6K tokens per model, compared with 2.93M and 3.71M tokens for those target subsets. Both target costs are measured from the runs underlying the fixed-panel scores, covering all 589 ExCyTIn questions and 25 CTI-REALM tasks. Figure 3 uses this same accounting across 14 budgets ($C = 10$–200), matching each proxy correlation to the mean target-subset correlation over 10,000 draws. Token accounting includes solver, benchmark judge where applicable, cached input and reasoning tokens. Table 1 retains the fixed $C = 100$ comparison.

Table 10: One-time token ledger and its role in the target-blind operating point. Target-benchmark runs validate the method but are not inputs to blind selection.

Component	Tokens	Share	Needed for blind selection?
Taxonomy, synthesis and item validation	31M	3.7%	yes
Fixed-panel bank grades	49M	5.7%	yes
ExCyTIn and CTI-REALM validation runs	774M	90.6%	no
Full research ledger	854M	100%	—

These marginal comparisons exclude the one-time 80M-token bank setup cost (31M for construction and 49M for grading). At $C = 25$, evaluating one additional model on both full targets costs 59.5M tokens, compared with 0.039M for the shared proxy. Under this full-target comparison, the setup cost is recovered after two additional models; including the 774M tokens used for target validation raises the research ledger to 854M tokens and the break-even point to 15 models. This calculation assumes reuse of the fixed bank and is distinct from the matched-correlation comparison above. Token counts do not directly measure monetary cost or wall-clock latency.

D LIMITATIONS

The model panel is small: with $n = 13$, individual rank changes materially affect ρ, and paired intervals for pool differences are wide. ExCyTIn and CTI-REALM are in-domain by construction, so their results do not identify trajectory provenance as a causal factor. SOCBench removes construction overlap but gives only marginal target-free transfer. The question-form ablation changes the mixture of items, not one form at a time, and supports a two-condition comparison rather than a claim that every form helps. Family holdout uses only eight predefined groups, four of which are singletons. Finally, capability load and provenance auditing each rely on one judge model, and single-shot items remain exposed to proxy gaming and distribution shift.