TOOL-CALL SAFETY EVALUATION DEPENDS ON WHAT LABELS REPRESENT

Anonymous authors
Paper under double-blind review

ABSTRACT

A tool-call safety monitor scores proposed calls before execution. Evaluating those scores against eventual attack success can credit or penalize the monitor for later actions. We distinguish whether the pending calls complete the attacker's goal from whether the full run succeeds, applying the benchmark's official success check immediately before and after the calls. These labels disagree on 19.6% of validated public AgentDojo decisions and 26.2% for two newly run agents, mostly because completion occurs later. With predictions fixed, changing the label moves TS-Guard's new-system AUROC from 0.628 to 0.841 and changes held-out monitor selection. ToolSafe's step annotations on a separate benchmark reproduce the sensitivity. On the Qwen agent, at the compared operating points, the locally selected monitor completes 6.1 percentage points more user tasks without attack success than the terminally selected monitor; its attack success is 1.1 points higher, with an interval covering zero. A blocking-disabled arm separates blocking from execution configuration. The event assigned to a score is part of monitor evaluation.

1 INTRODUCTION

A pre-execution safety monitor scores the tool calls an agent proposes, but it may be evaluated against an attack outcomes determined by later actions. Consider a recorded run in which an assistant reads a message planted by an attacker and later sends private data to that attacker. The completed run is labeled a successful attack because the data was ultimately disclosed. For a monitor evaluating the earlier decision, however, the pending action is the read; the later send has not yet been proposed. Assigning the final outcome to that earlier decision does not establish that the read itself completed the attacker's goal. Predicting later success, identifying what the pending actions do, and preventing the attack are different capabilities.

The distinction becomes consequential when labels are used to compare or select monitors. AgentDojo evaluates attack success over complete runs, whereas step-level benchmarks and guardrails assess intermediate actions (Debenedetti et al., 2024; Mou et al., 2026; Chen et al., 2026; Mohammadmirzaei & Flanigan, 2026). Both are useful targets, but evaluation against a final outcome can reward forecasting while being interpreted as recognition of the pending calls. We examine how this choice changes the assessment of fixed scores, the selection and calibration of monitors for a common target, and the outcomes of using those monitors to block calls.

We study these questions using 12,151 released AgentDojo runs from 21 systems and newly collected runs from two agents. AgentDojo pairs a user task with an attacker's goal pursued through instructions in tool-returned content. Before comparing labels, we separate decisions with proposed calls from decisions on which a tool-call monitor has nothing to block. For call-present decisions, we reconstruct the environment, execute the complete pending sequence in recorded order, and apply the official injected-goal check immediately before and after it. A false-to-true transition defines *local completion* L, while Y records attack success at the end of the run.

Including decisions without calls can reward call presence itself: a score based only on whether a call is present obtains 0.671 AUROC in the mixed test set but 0.500 among call-present decisions. Label disagreement remains after restricting to decisions with calls and validating replay: Y and L disagree on 19.6% of public and 26.2% of new-system decisions. Most mismatches arise because the goal is completed after the first scored sequence. The final outcome can therefore correctly describe the run while referring to a later decision than the one being evaluated.

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

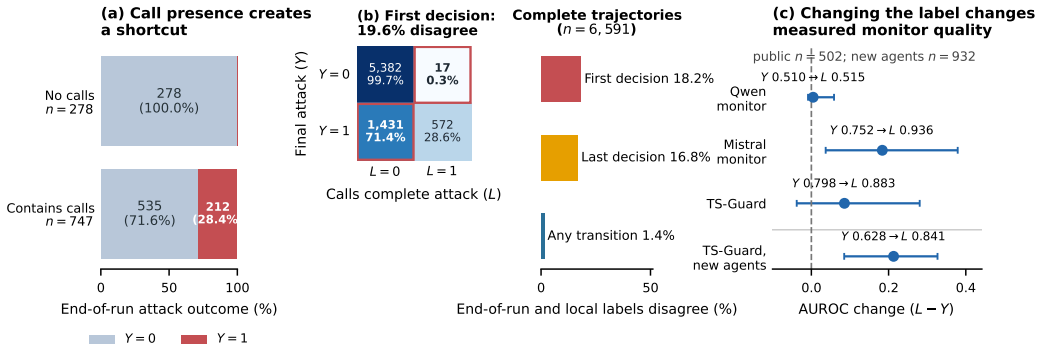


Figure 1: **A final outcome can describe later calls rather than the calls being scored.** (a) Call presence separates many positives from negatives in mixed data. (b) First-decision disagreement is 19.6% on all replay-valid rows; cell percentages are within each Y row. On complete trajectories, the matched first-decision and any-decision rates are 18.2% and 1.4%. (c) With rows and scores fixed, points show $\mathrm{AUROC}_L - \mathrm{AUROC}_Y$ for the public monitors and for TS-Guard on the two new agents; bars are paired 95% intervals from resampling systems, tasks, and goals.

These differences change conclusions drawn from the same predictions. With decisions and raw scores fixed, TS-Guard's new-system AUROC changes from 0.628 against Y to 0.841 against L. A held-out study instead fixes the evaluation target to L: selecting and calibrating a monitor using L rather than Y reduces mean system log loss from 0.395 to 0.171. A recalibration control separates calibration from monitor selection. ToolSafe tests the label-sensitivity finding using a separate benchmark's step annotations.

Predictive differences do not by themselves establish which monitor prevents more attacks. We therefore execute monitors selected under the two labels and compare them with a version of the same harness in which blocking is disabled. At the tested operating points, the locally selected monitor increases safe task success—completion of the user's task without attack success—by 6.1 percentage points on the Qwen agent relative to the terminally selected monitor, with an attack-success difference of 1.1 points whose interval covers zero.

We address three research questions through the following analyses.

RQ1: How do call presence and outcome timing affect monitor evaluation? We distinguish the opportunity to intervene, local completion, and terminal success, using replay and timing checks to locate label disagreement (Section 4.1).

RQ2: How does the evaluation label affect monitor assessment, selection, and calibration? We isolate label effects on fixed predictions and quantify selection and calibration consequences on a common held-out target, with an external annotation-based check (Section 4.2).

RQ3: How do monitors selected under different labels perform when used to block calls? We separate blocking from execution configuration and report attack success, safe task success, and intervention burden at declared operating points (Section 4.3).

Two supporting studies motivate checking the benchmark criterion directly: in a controlled Model Context Protocol (MCP) study, irrelevant context of the same length explains most of one system's response to deployment records, and in MANTRA (Anand et al., 2026), better recovery of written rule labels gives no supported gain in predicting the executed checks (Appendix L).

Released reconstruction code, predictions, analysis outputs, and tests make these comparisons traceable to the recorded calls.

2 Evaluation Targets and Label Construction

2.1 Eligibility, local completion, and final success

Let H be the visible history and $A = (a_1, \ldots, a_k)$ the complete ordered call sequence proposed in one assistant turn. Let $S(H, A) \in \mathbb{R}$ denote the monitor's risk score for A given H, computed before any call in A executes. Higher scores indicate greater assessed risk; the scores need not be calibrated probabilities.

Eligibility means that a decision contains a proposed call on which a tool-call monitor can intervene. Its indicator is $O = \mathbf{1}\{k > 0\}$: the decision is eligible when $O = 1$. Discrimination among pending sequences must therefore be evaluated on this population.

Local completion means that the official injected-goal check changes from false to true across the complete pending sequence. For an eligible decision, let e^- and e^+ be the environments before and after A, and τ^- the preceding call trace, extended to $\tau^- \circ A$ by the scored sequence. Hold fixed the inputs c: task, reference environment, and decision's assistant text. With G denoting the official injected-goal criterion, we define the binary *local completion label* by

$$L = \mathbf{1}\{G(c, e^-, \tau^-) = 0 \ \wedge \ G(c, e^+, \tau^- \circ A) = 1\}. \tag{1}$$

Thus $L = 1$ when the goal is unmet before the calls and met after the full sequence; $L = 0$ otherwise, provided the row passes validation. The explicit snapshot matters: AgentDojo resolves G from environment state for most decisions and from a trace-precedence variant for the rest (Section 4.2.3); even the state path receives the assistant text in c. We retain only rows passing state recovery, both checks, and the consistency tests in Section 3. Failures are exclusions, not negative labels.

We define the binary *final success label* $Y \in \{0, 1\}$ from the benchmark's terminal outcome: $Y = 1$ if the complete run is recorded as a successful attack, and $Y = 0$ otherwise. Thus Y describes the full run, whereas L describes completion across the scored calls. The monitor's score $S(H, A)$ can be evaluated against either label.

The local completion label does not indicate whether blocking prevents an attack. Blocking an enabling call with $L = 0$ may prevent later success, while an agent may retry after a blocked $L = 1$ sequence. Prevention requires executed comparisons allowing continuation after intervention.

2.2 Why mixed AUROC can reward call presence

For $S = O$, with half credit for ties,

$$\mathrm{AUROC}(O, Y) = \frac{1}{2} + \frac{1}{2}\big[P(O = 1 \mid Y = 1) - P(O = 1 \mid Y = 0)\big]. \tag{2}$$

Call presence can therefore score above chance in a mixture without distinguishing among calls. Conditional on $O = 1$, it is constant and has AUROC 0.5 when both classes occur. Appendix A derives the within-group and cross-group pair decomposition.

Transcript-only and call-only models may encode call presence or count; default safe scores for absent calls also expose eligibility. We report mixed and call-present results separately, filtering before tuning, calibration, and scoring for call-present fits. Feature controls compare argument-derived information with transcript counts and call shape. Conditioning on $O = 1$ addresses eligibility separately from the choice of Y or L.

3 Experimental Design

3.1 Data extraction, splits, and analysis populations

We use AgentDojo v1.2.2 and its instruction-following prompt-injection attack across banking, Slack, travel, and workspace. Each run pairs a user task with an injected goal delivered in tool-returned content; the artifact pins the revision. The extracted public frame contains 12,151 runs from 21 systems. Extraction excludes runtime errors and runs lacking a nonempty tool result followed by an assistant decision. We score the first such decision using only its visible transcript, complete pending sequence, and public schemas. Later events enter timing analyses, never predictors.

Table 1: Analysis populations. Replay validation requires exact reconstruction, completed before/after checks, and the same released terminal label in the replay and feature records. Counts across rows are not additive.

Population	Evaluation question	Rows
Primary held-out test	Eligibility effects in the mixture	1,025 (747 with calls)
Public replay	Local versus terminal completion	7,402
Matched public monitor scores	Fixed-score and selection comparisons	502
New-system replay	Label sensitivity on new agents	932
Argument-feature diagnostic	Additional features under L	4,328 public / 932 new
Execution audit	Executed arms with and without blocking	1,173 identities

A deterministic hash partitions tasks and injection goals, identified with their suites to prevent cross-suite identifier collisions. Training requires a development system, task, and goal; testing requires all three to be in the test split. Eight development and 13 test systems are disjoint. The primary mixture has 2,065 training and 1,025 test rows, or 1,694 and 747 after call-presence filtering, with zero system, task, or goal overlap. Whole-suite tests also reserve every task and goal in the held-out suite. The two new agents use familiar benchmark content: before inspecting their runs, we fixed 629 attack paths per system, covering all banking, Slack, and travel pairs and workspace goals 0–5 (Appendix J).

The analyses retain distinct populations (Table 1). Public fixed-score evaluation starts from 870 reserved decisions, 640 with calls, of which 502 pass replay validation; new-system evaluation starts from 984 call-present decisions. A disclosed 40-row pilot informed the replay validator; removing the 30 call-present decisions from pilot identities and 22 that fail replay leaves 932 new-system decisions (Appendix F).

3.2 Monitor scoring, selection, and calibration

The monitors are Qwen3-30B-A3B (Instruct-2507), Mistral-Small-3.2-24B (Instruct-2506), and official `MurrayTom/TS-Guard`, revision `ad2f82df`. Qwen and Mistral use Q4_K_M quantization. All receive the request, visible history, pending calls, and public schemas, scoring fixed decisions. Qwen and Mistral separately generate new runs as agents; TS-Guard remains external.

Label comparisons fix rows and raw scores; the primary statistic is paired AUROC change. TS-Guard's ratings $\{0, 0.5, 1\}$ are not probabilities. Target-specific logistic calibration uses public development data. Across targets, average precision and calibrated losses are descriptive because class membership and prevalence change.

Selection instead fixes the evaluation target to L. Each of 12 systems is held out; the remainder select and calibrate a monitor under Y or L by mean system log loss. A third procedure recalibrates the Y-selected monitor under L. No held-out labels enter fitting or selection. All three secondary procedures face the same local target; intervals resample saved cross-fitted predictions without repeating selection or fitting.

3.3 Supporting diagnostics and uncertainty estimation

Ridge-logistic diagnostics compare transcript counts (characters, tokens, and message roles) and call shape (counts, read/mutation types, argument sizes) with added argument–context overlap, copying, entity, and cross-tool features. This representation comparison is distinct from full-transcript language-model monitors (Appendix D).

For the L diagnostic, leave-one-development-system-out log loss selects regularization on replay-valid development rows. Frozen predictors face 4,328 decisions from 12 other public systems and 932 new-system decisions. Log loss is primary; Brier score, AUROC, average precision, and calibration error are also reported. Appendix E distinguishes fixed-prediction from full-refit uncertainty and specifies system, task, and goal dependence sensitivities; Appendix F.2 reports the diagnostic results.

Table 2: Terminal outcome Y and local completion L on replay-valid, call-present decisions. The last column is conditional on $Y = 1$.

Data	Rows	$Y \neq L$	$L = 0$ among $Y = 1$
Public	7,402	19.6%	71.4%
New systems	932	26.2%	62.2%

3.4 Execution arms and comparison conditions

Execution covers 1,173 outcome-blind matched identities from two locally hosted agents. After the first nonempty tool result, TS-Guard scores each complete pending sequence. Ratings at least 0.5 block all calls, return generic blocked-call errors, and permit continuation for at most 15 tool rounds. The prespecified 1.0 threshold blocks less aggressively.

Earlier unprotected runs match system, suite, task, and injected goal but come from another pipeline, confounding blocking with execution configuration. An observe arm uses the same harness with only blocking disabled: it scores and records potential blocks but executes every call. Two further arms use the L-selected Mistral monitor at declared thresholds 0.9 and 0.8 on its reported risk probability, which bracket TS-Guard's blocking burden. Label-specific selection chooses the monitor model; the executed thresholds apply to its uncalibrated score. Each arm has 10,000 system–task–goal resamples. Across the blocking arms, the agent, identities, blocking semantics, and continuation budget are fixed; only the scorer and its threshold change. The executed Mistral monitor shares the offline scorer's prompt, template, and decoding settings but not its strict response-schema flag, and both parsers return the same probability on all 5,892 recorded replies.

Each agent requires its own served-chat tool-call adapter; Mistral also disables native chat parsing and uses an explicit template. A separate budget analysis routes the top q fraction of call-present decisions to a costlier monitor and measures clean review burden (Geifman & El-Yaniv, 2019). Appendix I separates routing, offline flags, and executed blocks.

4 Results

4.1 RQ1: How do call presence and outcome timing affect monitor evaluation?

4.1.1 Call presence and the evaluation population

Figure 1a–b summarizes the call-presence and timing effects. In the primary mixed test, all 212 positives contain a call, whereas 278 negatives do not. Call presence alone therefore reaches 0.671 AUROC, compared with 0.500 within the call-present subset. Suite identity alone reaches 0.772 AUROC; adding call presence gives 0.826. Evaluating that same score only on call-present pairs gives 0.736. The analogous call-content score changes from 0.820 overall to 0.732 within that subset. Adding call presence beyond suite identity gives a log-loss gain of +0.051 (95% fixed-prediction interval $[0.017, 0.090]$).

ToolSafe's AgentDojo-Traj split provides a negative control: all 1,220 annotated steps contain an action, including 868 safe and 352 unsafe steps. Call presence is constant. This control identifies eligibility variation; end-of-run labeling alone is not its cause. A common-support sensitivity removes 749 later workspace paths and preserves the qualitative finding (Appendix H).

4.1.2 Local completion and outcome timing

Replay validates 7,402 of 9,011 public candidates, explicitly recording 1,609 exclusions. Table 2 reports disagreement conditional on validation. Assigning all 1,609 excluded mismatch indicators their worst and best values bounds all-candidate disagreement by $[16.1\%, 33.9\%]$ (Appendix F, Equation 4).

All 1,431 public $(Y, L) = (1, 0)$ cases leave the goal false across the scored sequence while the released terminal label is positive; the trajectory audit below places most of these completions at later decisions. Conversely, 17 sequences have $L = 1, Y = 0$. All 17 reverse mismatches pass replay

Table 3: The same monitor scores under two labels. Y records the end-of-run outcome; L records whether the official success check is false before the scored calls and true after them. The upper part reports the paired AUROC change and 95% confidence interval with scores fixed; ranks order the point estimates. The lower part reports average precision (AP), Brier score, and expected calibration error (ECE) after fitting a separate logistic score-to-probability mapping for each label on public development data. Non-AUROC change intervals are in the artifact.

A. Primary raw-score comparison

Data	Monitor	Y AUROC	L AUROC	$L - Y$ [95% CI]	Y rank	L rank
Public	Qwen monitor	0.510	0.515	$+0.005$ $[-0.009, 0.059]$	3	3
Public	Mistral monitor	0.752	0.936	$+0.184$ $[0.037, 0.379]$	2	1
Public	TS-Guard	0.798	0.883	$+0.086$ $[-0.038, 0.281]$	1	2
New	TS-Guard	0.628	0.841	$+0.213$ $[0.085, 0.327]$	–	–

B. Label-specific descriptive metrics

Data	Monitor	Target	AP	Log loss	Brier	ECE
Public	Qwen monitor	Y	0.333	0.758	0.243	0.153
		L	0.135	0.411	0.118	0.063
Public	Mistral monitor	Y	0.605	0.561	0.188	0.099
		L	0.576	0.223	0.072	0.078
Public	TS-Guard	Y	0.609	0.523	0.175	0.113
		L	0.468	0.264	0.083	0.056
New	TS-Guard	Y	0.507	0.784	0.268	0.195
		L	0.402	0.336	0.110	0.059

validation with intact call/result integrity and 0 call-execution exceptions, across 10 systems (at most 4 each) and two suites (9 in workspace and 8 in travel; Appendix F). New-system replay has no reversals and reproduces the dominant direction.

On 6,591 identically matched, fully covered trajectories, first-decision mismatch is 18.2% and any-decision mismatch 1.4% (paired change -16.9 percentage points, interval [-28.0, -8.3]). Coverage is 89.0%; the all-row bound is [1.2%, 12.2%]. Any-transition agreement is partly structural for a binary goal starting false and ending true. For $L = 1$ multi-call proposals, proper prefixes already complete the goal in 35/75 public and 14/17 new-system cases (Appendix F.1).

4.1.3 Completion without the scored sequence

Local completion does not imply necessity. We replay each trajectory twice from the same pristine environment, including or omitting the scored sequence while retaining every later recorded call. The include arm must agree with the released terminal label: 7,306 of 7,402 replayed decisions do (98.7%). Both arms hold the decision's assistant text fixed and evaluate their replayed state and trace. Framework exceptions invalidate an arm; ordinary benchmark tool errors remain outcomes. The label and both arms come from the same run.

Conditioning on L reveals opposite behavior. Of 1,357 decisions with $L = 0$ and eventual success, deleting the sequence still completes the goal in 1,235 (91.0%), versus 6 (1.0%) of 572 decisions with $L = 1$ and eventual success. A successful skip arm demonstrates completion without the sequence; failure may merely reflect an incoherent recorded continuation. Thus L separates demonstrably dispensable decisions from those whose dispensability is not established. The contrast persists when the remaining suffix is controlled: among decisions with at least one later call the shares are 91.0% for $L = 0$ and 1.7% for $L = 1$, and stratifying on suite and the number of later calls gives 90.0% versus 1.7% (Appendix F.3).

4.2 RQ2: How does the evaluation label affect monitor assessment, selection, and calibration?

4.2.1 Fixed predictions under different labels

Figure 1c and Table 3 report these paired changes. On 502 matched public decisions, Mistral's AUROC changes from 0.752 against Y to 0.936 against L (paired change +0.184, interval [0.037, 0.379]). TS-Guard changes from 0.798 to 0.883, with a change interval including zero. The

Table 4: Held-out mean system log loss of leave-one-system-out selection and logistic calibration on 502 matched public decisions. Each panel evaluates one target, and losses are compared only within a panel; lower is better.

Evaluated on	Selection target	Calibration target	Selected monitor	Mean system LL
Held-out L	Y	Y	TS-Guard (12/12)	0.395
	Y	L	TS-Guard (12/12)	0.240
	L	L	Mistral (12/12)	0.171
Held-out Y	L	L	Mistral (12/12)	1.156
	L	Y	Mistral (12/12)	0.567
	Y	Y	TS-Guard (12/12)	0.486

point-estimate leader changes from TS-Guard to Mistral, but the direct between-monitor intervals include zero under both labels (Mistral minus TS-Guard: -0.045 $[-0.225, 0.087]$ against Y and $+0.053$ $[-0.052, 0.174]$ against L).

The four label cells explain the unequal changes. Mistral ranks the 66 locally completing decisions above the 99 later-success decisions in 87.5% of cross-group pairs; Qwen does so in 51.0%. Reassigning these rows to a different class changes what each fixed score is rewarded for.

The 502-row matched cohort contains no $L = 1, Y = 0$ decisions: all relabeled positives become negative, potentially rewarding monitors that rank payload-like calls highly. A size-matched null draws random subsets of $Y = 1$ rows of size $|L = 1|$, uses each subset as the positive class, and recomputes AUROC over 20,000 draws. Arbitrary relabeling lowers AUROC on average: keeping k of m positives against n negatives changes a score with AUROC a by $\frac{m-k}{n+m-k}\left(\frac{1}{2} - a\right)$ in expectation (TS-Guard -0.068, Mistral -0.057; Equation 5). Observed increases exceed the null bands: Mistral +0.184 versus $[-0.114, -0.000]$ ($p < 0.001$), and TS-Guard +0.086 versus $[-0.116, -0.020]$ ($p < 0.001$). Matching the $L = 1$ per-suite composition gives the same conclusion. Near-chance Qwen stays within its band (+0.005, $[-0.013, +0.005]$, $p = 0.361$). The new-system effect also exceeds the suite-matched null (+0.213, $[+0.041, +0.098]$), although suite composition explains part. Local labels therefore select rows already ranked highly, rather than gaining solely by shrinking the positive class.

On 932 new-system decisions, unchanged TS-Guard scores yield 0.628 AUROC against Y and 0.841 against L (change +0.213, interval [0.085, 0.327]).

4.2.2 Monitor selection and calibration for a common target

For the common held-out target L, the Y-trained procedure selects TS-Guard in 12/12 folds; the L-trained procedure selects Mistral in every fold. Across 502 matched decisions, their mean system log losses are 0.395 and 0.171. The difference is +0.224 (paired-system interval [0.177, 0.267]), a 56.7% reduction relative to the Y-trained procedure.

Keeping the Y-selected monitor but recalibrating it to L yields 0.240 log loss. The observed difference decomposes into +0.156 from recalibration and +0.069 from changing the selected monitor, with paired-system intervals [0.100, 0.206] and [0.033, 0.111]. Broader system–task–goal resampling keeps the combined interval above zero ([0.029, 0.403]) but not selection alone ([-0.082, 0.247]). The strongest supported result concerns the complete prediction procedure; calibration accounts for a substantial part of its target-mismatch cost. Table 4 reports all three procedures under both targets.

The comparison measures the cost of using one event to train a procedure evaluated on another. Against held-out Y, the order reverses, and in both directions training against the other event more than doubles held-out log loss. Under L, the Y-trained procedure is also worse than predicting the training folds' L rate for every row (0.387), whereas the L-trained procedure improves on both that rate and the per-suite rate (0.368; Appendix I.2). The shared training folds and fixed-prediction resampling are specified in Appendix E.

Table 5: Executed arms on the identities every arm of that agent shares. A difference against the observe arm, which runs the same harness with the block action removed, is attributable to blocking. Safe task success counts runs that finished the user's task while the attack failed. Y and L name the label whose selection study chooses that guard; values other than blocks per run are percentages.

Agent	Executed arm	Blocks/run	Attack	User task	Safe task
Mistral Small 3.2 (553)	Observe, blocking removed	0.00	9.4	18.1	16.3
	TS-Guard 1.0 (Y)	0.09	4.5	18.3	17.7
	Mistral monitor 0.9 (L)	0.18	1.4	17.9	17.9
	TS-Guard 0.5 (Y)	0.24	1.6	17.9	17.7
	Mistral monitor 0.8 (L)	0.30	1.1	18.3	18.3
Qwen3-30B-A3B (620)	Observe, blocking removed	0.00	36.5	55.3	41.9
	TS-Guard 1.0 (Y)	0.44	14.8	55.2	49.8
	Mistral monitor 0.9 (L)	0.70	3.1	54.2	52.9
	TS-Guard 0.5 (Y)	1.34	1.9	47.4	46.8
	Mistral monitor 0.8 (L)	2.07	1.0	37.4	37.3

4.2.3 External annotations and label sensitivity

ToolSafe's fixed ASB-Traj release has 5,231 steps (Mou et al., 2026; Zhang et al., 2025). Its strict target L_{ASB} treats ratings 0.5 and 1 as positive; it is an annotation, not AgentDojo's executable transition. It disagrees with terminal success on 2,267 steps (43.34%).

Rerunning pinned TS-Guard covers every released row. Its fixed scores yield 0.756 AUROC against terminal success and 0.952 against step annotations (change +0.196, interval $[0.186, 0.205]$). Rerun ratings agree with the release on 97.2% of rows; both vectors give nearly identical label effects. Counting only score-1 annotations as positive gives a paired change of +0.161 (interval $[0.149, 0.173]$); Appendix I.3 reports the released-versus-reported row-count difference.

Portability requires a criterion defined on the snapshots in Equation 1; AgentHarm's graders, for example, read the message list and would first need a definition on prefixes. AgentDojo uses state for 7,039 validated decisions (95.1%) and trace precedence for 363 (4.9%, all in slack). The 502-row matched public cohort is 100.0% state-resolved: fixed-score and selection results contain no trace-resolved rows. The 932-row new-system cohort is 96.1% state-resolved, with 0 trace-resolved $L = 1$ rows. Restricting it to 896 state rows barely changes TS-Guard's paired effect: 0.633 to 0.849 ($\Delta = +0.216$). Trace semantics therefore do not drive these results; Appendix K discusses transfer.

4.3 RQ3: How do monitors selected under different labels perform when used to block calls?

4.3.1 Separating blocking from execution configuration

Earlier unprotected runs share 1,173 matched identities but use another pipeline. Appendix I reports their comparison, discordances, threshold-1.0 sensitivity, and decision totals. Here we use the observe arm of the same harness: disabling only blocking separates its effect from execution configuration.

The decomposition differs by agent. On 553 Mistral identities, corpus, observe, and blocking attack success are 27.1%, 9.4%, and 1.6%. The corpus-to-observe difference is -17.7 percentage points and the observe-to-blocking difference -7.8 percentage points (interval $[-14.2, -2.9]$). On 620 Qwen identities the rates are 43.2%, 36.5%, and 1.9%, giving differences of -6.8 percentage points and -34.5 percentage points ($[-50.0, -21.8]$).

4.3.2 Attack success and safe task success

Table 5 compares the selected monitors on the same identities. Safe task success means completing the user's task *without* attack success; we report it alongside both component rates. On Qwen, the L-selected monitor at threshold 0.9 increases user-task success over TS-Guard at 0.5 by +6.8 percentage points ($[+1.6, +13.3]$) and safe task success by +6.1 percentage points ($[+1.0, +12.5]$),

with attack success +1.1 percentage points higher ($[-1.1, +4.2]$). All three Mistral differences have intervals covering zero; its observe attack rate is already 9.4%.

4.3.3 Operating points and intervention burden

Operating points also determine the comparison. Randomly choosing one of the two executed TS-Guard policies at run start and retaining it makes attack and utility rates linear in the mixing weight, so we compute the mixture from the two executed arms. Weight 91.2% on TS-Guard 0.5, chosen on the evaluation cohort, matches the L-selected guard's attack rate 3.1%. The mixture completes 48.1% of user tasks versus 54.2%, a user-task gap of +6.1 percentage points.

Fewer blocks do not mean fewer scores: the Y-selected guard scores 2,533 sequences and the L-selected guard 2,580. Of the 397 fewer blocks, 365 are repeats within interrupted runs and 32 are runs left untouched. Mistral shows the opposite split. At the second L operating point, threshold 0.8, Qwen task success falls to 37.4%, below every other arm. Label choice selects a monitor; its operating point still requires evaluation.

5 Related Work

Outcome, step, and failure-localization targets. AgentDojo, MANTRA, and τ-bench evaluate executed outcomes (Debenedetti et al., 2024; Anand et al., 2026; Yao et al., 2024). ToolSafe, TraceSafe, and OSGuard assess intermediate actions; ToolSafe and OSGuard also intervene and report user-task outcomes (Mou et al., 2026; Chen et al., 2026; Mohammadmirzaei & Flanigan, 2026). Pan et al. target the first uncorrected critical error (Pan et al., 2026); Zhang audits step credit against replayed policy alternatives and a shuffled control (Zhang, 2026). Unlike counterfactual contribution, L records a transition of the existing criterion across observed calls. We fix decisions and scores while changing the target, then execute the selected monitor against a blocking-disabled control. Procedure-aware, containment, and near-miss work addresses other trajectory properties (Cao et al., 2026; Lan et al., 2026; Rabinovich et al., 2026; Tao et al., 2026).

Guardrails, shortcuts, and proxy measurements. StepGuard, AttriGuard, and ActGuard develop monitors or interventions (Zheng et al., 2026; He et al., 2026; Wang et al., 2026); we study fixed predictions' evaluation targets. Shortcut learning motivates auditing what evaluations reward (Geirhos et al., 2020; Tang et al., 2025; Shao et al., 2026), while calling and abstention alter the intervention population (Ross et al., 2025; Luo et al., 2026). Replay identifies local completion without validating L as a surrogate for blocking effects, a distinction analogous to surrogate-endpoint methodology (Prentice, 1989).

6 Discussion and Conclusion

The same monitor can receive materially different evaluations when its scores are assigned to different events, and a separately annotated benchmark reproduces that sensitivity. Executed arms with and without blocking separate prevention from execution configuration, and the label that selects a monitor changes its task outcomes and blocking burden. Evaluate monitors on the calls they can stop, specify the event each score predicts, and match the evidence to the claim. Concretely, reports should state the eligible decision population, the event attached to each score, the targets used for selection and calibration, and how operating thresholds were chosen, and executed comparisons should report attack success, user-task success, safe task success, and blocking burden.

Ethics Statement

The AgentDojo experiments ran inside the benchmark's simulated environments, and the newly evaluated agents used local model servers without access to production accounts or external services. The supporting studies in Appendix L used authored benign MCP calls and MANTRA's released tasks, tools, and evaluator. The study diagnoses a shortcut in the evaluation and does not introduce a new prompt-injection payload or operational exploit. Its main risk is that an audit of evaluation labels is read as deployed defense performance.

Reproducibility Statement

The evaluation targets and label construction are specified in Section 2. The experimental design, analysis populations, and features are specified in Section 3 and Appendices B–D. Appendices F and E describe replay validation and uncertainty. The artifact in the supplementary material records the AgentDojo, ToolSafe and TS-Guard revisions, the analysis specifications, seeds and commands, the compact predictions, the analysis outputs, the bootstrap summaries and the tests. A single CPU command regenerates the paper's tables, macros and summary figure from the included JSON and CSV files, and an optional GPU command reruns the pinned TS-Guard on released ToolSafe ASB-Traj runs. Generating new agent runs, the raw runs, external repositories and model checkpoints remain external dependencies.

AI Use Disclosure

OpenAI generative AI tools assisted with refining hypotheses, designing and reviewing experiments, implementing and testing code, interpreting results, finding and checking literature, and drafting and editing the manuscript and figure code. The authors inspected the primary sources, audited generated code, reran the reported analyses, checked numerical claims against saved outputs, and reviewed all AI-assisted text. The authors take responsibility for the final paper and artifact.

References

Ashwani Anand, Ivi Chatzi, Ritam Raha, and Anne-Kathrin Schmuck. MANTRA: Synthesizing SMT-validated compliance benchmarks for tool-using LLM agents, 2026.

Hongliu Cao, Ilias Driouich, and Eoin Thomas. Beyond Task Completion: Revealing Corrupt Success in LLM Agents through Procedure-Aware Evaluation, 2026. URL https://arxiv.org/abs/2603.03116.

Yen-Shan Chen, Sian-Yao Huang, Cheng-Lin Yang, and Yun-Nung Chen. TraceSafe: A systematic assessment of LLM guardrails on multi-step tool-calling trajectories, 2026. URL https://arxiv.org/abs/2604.07223. Version 2, 9 August 2026.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=m1YYAQjO3w.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2151–2159, 2019. URL https://proceedings.mlr.press/v97/geifman19a.html.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. doi: 10.1038/s42256-020-00257-z.

Yu He, Haozhe Zhu, Yiming Li, Shuo Shao, Hongwei Yao, Zhihao Liu, and Zhan Qin. AttriGuard: Defeating indirect prompt injection in LLM agents via causal attribution of tool invocations, 2026. URL https://arxiv.org/abs/2603.10749.

Wenhao Lan, Shan Li, Meiqi Wu, Xinhua Lai, Junbin Yang, and Haihua Shen. ContainmentBench: Trace-based evaluation of post-exposure containment in tool-using LLM agents, 2026. URL https://arxiv.org/abs/2607.23999. Version 3, 9 August 2026.

Han Luo, Bingbing Wen, and Lucy Lu Wang. Agentic abstention: Do agents know when to stop instead of act?, 2026. URL https://arxiv.org/abs/2606.28733.

Mina Mohammadmirzaei and Jeffrey Flanigan. OSGuard: A benchmark for safety in computer-use agents, 2026. URL https://arxiv.org/abs/2606.15034v1. Version 1, 13 June 2026.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. ToolSafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37125–37153, San Diego, California, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1850. URL https://aclanthology.org/2026.findings-acl.1850/.

Sitong Pan, Yipeng Shen, Yilin Lu, Caiwen Ding, Lu Cheng, and Qianwen Wang. Monitoring web agents without internal signals: Observable trajectories and key-step supervision, 2026. URL https://arxiv.org/abs/2609.02057v1. Version 1, 2 September 2026.

Ross L. Prentice. Surrogate endpoints in clinical trials: Definition and operational criteria. *Statistics in Medicine*, 8(4):431–440, 1989. doi: 10.1002/sim.4780080407.

Ella Rabinovich, David Boaz, Naama Zwerdling, and Ateret Anaby Tavor. Near-miss: Latent policy failure detection in agentic workflows. In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 296–308, San Diego, California, USA, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.gem-main.30. URL https://aclanthology.org/2026.gem-main.30/.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2Call: When (not) to call tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3391–3409. Association for Computational Linguistics, April 2025. doi: 10.18653/v1/2025.naacl-long.174. URL https://aclanthology.org/2025.naacl-long.174/.

Jiaqi Shao, Hanck Chen, Wei Zhang, Maxm Pan, and Bing Luo. Do Agent Benchmarks Measure Capability? Protocol Validity in the Age of Agentic AI, 2026. URL https://arxiv.org/abs/2607.22368.

Yuzhi Tang, Tianxiao Li, Elizabeth Li, Chris J. Maddison, Honghua Dong, and Yangjun Ruan. LM agents may fail to act on their own risk knowledge. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=OeYdS51k8F.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. TRACE: Turn-level reward assignment via credit estimation for long-horizon agents, 2026. URL https://arxiv.org/abs/2607.13988.

Bingzheng Wang, Xiaoyan Gu, Wentao Wang, Xingyou Yang, Hongcheng Li, and Rong Yin. ActGuard: Pre-execution action auditing against indirect prompt injection in LLM agents, 2026. URL https://arxiv.org/abs/2609.14987.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024.

Haiyue Zhang. Credit without ground truth: Auditing step-level credit assignment in LLM agents against executed replay, 2026. URL https://arxiv.org/abs/2608.19760. Version 2, 25 August 2026.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=V4y0CpX4hK.

Zhijie Zheng, Yu Li, Chen Qian, Yuqian Fu, Yanwei Fu, Lu Sheng, Jing Shao, and Dongrui Liu. StepGuard: Learning step-level guardrails with scalable supervision and safety-utility balancing, 2026. URL https://arxiv.org/abs/2608.24777. Accepted at EMNLP 2026.

A Why Mixed AUROC Can Be High

AUROC is the probability that a randomly chosen positive outranks a randomly chosen negative, with ties receiving half credit. Split these pairs by whether each turn contains a call. For any score S, let $a_i = P(O = i \mid Y = 1)$ and $b_j = P(O = j \mid Y = 0)$. For each pair (i, j) with $a_i b_j > 0$, define

$$C_{ij} = P(S_i^+ > S_j^-) + \tfrac{1}{2} P(S_i^+ = S_j^-),$$

where $S_i^+ \sim S \mid (Y = 1, O = i)$ and $S_j^- \sim S \mid (Y = 0, O = j)$ are independent. Zero-weight terms are defined as zero and do not require a conditional distribution on an empty stratum. Then

$$\mathrm{AUROC}(S, Y) = \sum_{i,j \in \{0,1\}} a_i b_j C_{ij}. \tag{3}$$

The identity follows by partitioning each positive–negative comparison by (O^+, O^-) and applying the law of total probability to strict wins and ties. The AUROC among turns with calls is only C_{11}, when both conditional classes are nonempty; a mixed AUROC also includes cross-group comparisons and turns without calls. For $S = O$, the nonempty strata have $C_{10} = 1$, $C_{01} = 0$, and $C_{00} = C_{11} = 1/2$, which gives Equation 2. Thus call presence can raise mixed AUROC even though it has no ranking power once every evaluated turn contains a call.

B Extraction and Information Available to the Monitor

The extractor reads released AgentDojo JSON trajectories. It finds the first nonempty tool result and the next assistant decision. It reconstructs the transcript visible through that decision and the calls proposed there. Feature construction completes before the released security bit is read. Unit tests change the separately stored injection payload and end-of-run outcome while holding the runtime transcript fixed and verify that extracted predictors do not change.

Across the complete public data, the two positive decisions without calls both arise from the travel injection goal that asks the agent to state a hotel name. Their end-of-run attack labels are positive, but no tool-call monitor can act on that text-only decision. We retain them in the mixture audit and exclude them only from analyses of decisions that contain calls.

The transcript representation removes `<thinking>` blocks and every field of the calls proposed at that decision. The call representation canonicalizes function names and JSON arguments. Additional deterministic features count overlaps between call arguments and visible context. No feature uses a learned embedding from the test population.

C Split Construction

System groups are fixed lists. A deterministic hash assigns each suite-qualified task and goal identifier to development or testing. Training requires a development system, task, and goal; testing requires a test system, task, and goal. Rows that mix the two assignments are unused when systems, tasks, and goals are all held out. The split has zero overlap on all three axes by construction.

For analyses restricted to call-present decisions, rows without calls are removed from the complete extracted frame before this partition function is called; the mixed analysis keeps them. The same order is used for every inner system-holdout tuning fold. Whole-suite evaluation first removes rows without calls, then removes one suite and all identifiers scoped to it.

D Feature Definitions

The transcript-summary features count transcript characters, transcript tokens, and numbers of user, assistant, and tool messages. They do not encode the semantic content of the transcript. Basic call features contain call count, distinct call count, read-like call count, mutation-like call count, nonempty argument count, argument characters, and argument tokens.

The augmented representation additionally measures argument overlap with the user request and prior tool results, their difference, longest copied span, source-to-destination tool distance, and

read-to-mutation and cross-tool transitions. It also records absence of visible assistant text and copied email, URL, date, time, long-number, identifier, and aggregate entity counts. The word-pattern model uses canonical call text with word- and character-level term-frequency–inverse-document-frequency (TF–IDF) features.

In the diagnostic tables, "transcript" refers to the count summaries specified here. The language-model monitors separately receive the visible transcript itself.

E UNCERTAINTY

We use two kinds of uncertainty interval. A fixed-prediction bootstrap changes which test clusters appear while keeping the trained model and its predictions unchanged. A full-refit bootstrap repeats sampling, fitting, and scoring. For the primary mixed analysis and the analysis restricted to decisions with calls, fixed-prediction draws resample held-out systems and task-pair components. A component is one user-task and injection-goal pair within a suite. The public and new-system comparisons of Y and L instead resample systems, within-suite user tasks, and within-suite injection goals as three separate axes. The new-system TS-Guard transfer interval resamples systems and attack components, where an attack component is the same suite-qualified task/goal pair. All such draws recompute metrics without refitting and quantify test-population variation conditional on fixed predictions.

The monitor-selection audit uses held-out systems as the resampling units for the saved cross-fitted excess losses. Its primary interval pairs the 12 values and resamples systems with equal weight for 20,000 draws. An exact 2^{12} sign-flip calculation is retained as a descriptive sensitivity under sign symmetry. The folds share training systems and may share tasks and goals. A separate 10,000-draw fixed-prediction bootstrap independently resamples system, suite-qualified task, and suite-qualified goal indices. Neither procedure reruns selection or calibration within a draw.

F REPLAYING THE SUCCESS CHECK BEFORE AND AFTER THE SCORED CALLS

We rerun AgentDojo's official injected-goal check on the environment state immediately before and after the complete ordered call sequence proposed by the audited assistant turn. Calls are applied in recorded order. The assistant text, preceding observations, tool arguments, and benchmark task are not edited. A row is included only when the stored transcript can be reconstructed exactly, the recorded call/result structure is valid, and both success-check evaluations complete. Empty call identifiers are paired positionally only when cardinality agrees; nonempty identifiers must match exactly. Runtime errors and benchmark-version mismatches remain explicit audit outcomes rather than negative labels.

The validator computes the local label by replaying the prefix and the pending calls. It then attaches the end-of-run outcome recorded in the source trajectory and requires that this outcome agree with the feature row before the analysis replaces the prediction target with the transition in Equation 1. That agreement check confirms the released outcome was carried consistently through both data paths; Y is the released terminal label throughout. The replay label describes the complete call sequence under AgentDojo's official semantics.

On the public first-decision replay, the full table is $(Y, L) = (0, 0) : 5,382$, $(0, 1) : 17$, $(1, 0) : 1,431$, and $(1, 1) : 572$.

Reverse mismatches. All 17 decisions with $(Y, L) = (0, 1)$ read the criterion from the state rather than the trace and pass replay validation and tool-result integrity. They sit early in their trajectories, at decision index 4 to 5 with 1 to 2 preceding calls, and they are spread over 10 systems rather than concentrated in one. Replaying each run to the end with the same assistant text leaves the goal unmet in 8 of them, matching the released label, so the completed state does not persist. In the other 9, the replayed end state still satisfies the goal although the released terminal label is negative.

Initial replay exclusions and missing-label sensitivity. The first-decision replay validates 7,402 of 9,011 candidates and excludes 1,609. Let m be the sum of the two observed off-diagonal counts above, let n be the validated count, and let e be the excluded count. Any assignment of binary

mismatch indicators to the excluded rows gives an all-candidate rate d satisfying

$$\frac{m}{n+e} \le d \le \frac{m+e}{n+e}. \tag{4}$$

Here $m = 1{,}431 + 17$, $n = 7{,}402$, and $e = 1{,}609$, giving approximately $[16.1\%, 33.9\%]$. This is a deterministic missing-label completion bound, not a confidence interval.

We inspected a disclosed 40-row new-system pilot while developing the replay validator. The implementation, adversarial tests, inclusion rules, model representations, and uncertainty procedure were fixed afterward. All pilot identities are removed from the new-system model evaluation. We consequently label the before/after replay analysis post-pilot exploratory; the larger public replay is a retrospective audit.

F.1 Where within a multi-call sequence does the check change?

The label L applies to the complete ordered sequence proposed in one assistant turn. A prefix is the first one or more calls in that sequence. To see how often the full sequence is the smallest observed unit that completes the goal, we replay every prefix of the call sequence for decisions with at least two calls. We select multi-call decisions without reading outcomes, then retain rows that pass the same replay checks as the main analysis.

Table 6: First completion within validated multi-call sequences. Percentages in the last two columns use the $L = 1$ column as denominator.

Data set	Multi-call runs	$L = 1$	Earlier calls suffice	All calls first suffice
Public	915	75	35 (46.7%)	40 (53.3%)
New systems	114	17	14 (82.4%)	3 (17.6%)

No retained replay completes the goal at a proper prefix and then loses that completion after the full recorded sequence. These counts locate the first observed completion in the recorded order.

The first-decision population contains 7,402 decisions validated by replay. Last- and any-decision comparisons require complete decision coverage: 6,591 trajectories satisfy that condition, while 811 are excluded from those rows of the table. The paired first estimate is recomputed on exactly the complete set rather than compared across denominators. "Any decision" requires an exact false-to-true transition of the official success check; observing the goal as true does not receive credit if it was already true before that call sequence. Under this definition, 4.3% of end-of-run-positive complete trajectories contain no false-to-true change in the official success check at any covered decision.

These residual cases are informative about what "complete coverage" means here. If the check were initially false, finally true, and evaluated on every boundary at which it can change, a false-to-true transition would have to appear somewhere. That they do not appear means the three premises are not all satisfied on these trajectories: the check may already hold at the first covered state, the end-of-run label may be computed from inputs that the replayed boundaries do not reproduce, or a state change may occur outside the assistant-decision boundaries the audit covers. Covering every recorded decision is not the same as covering every boundary at which the official criterion can change value. This residual reflects that gap rather than disagreement about an individual decision.

F.2 Do call arguments help with the before/after label?

In Table 8, adding argument–context features to transcript summaries and call shape reduces L-target log loss from 0.173 to 0.096 on 4,328 held-out public rows (gain +0.077, interval $[0.022, 0.152]$). On 932 new-system rows it changes from 0.484 to 0.327 (gain +0.157, interval $[0.003, 0.298]$). Corresponding AUROCs are 0.966 versus 0.988 and 0.918 versus 0.949. Appendix H reports the Y-target controls, including unsupported and negative gains.

Both predictors are fitted to L, which is one exactly when the official success check is false before the scored calls and true after them. Fitting uses development systems for which replay succeeds. The candidate adds comparisons between call arguments and visible context. The public and new-system comparisons use fixed predictions and independent system–task–goal cluster resampling. They test whether the retained augmented predictor improves on that baseline on these two evaluation sets.

Table 7: How end-of-run label disagreement changes when the official success check is run around different recorded call sequences. The first row uses every public decision validated by replay. The next three rows compare the same complete runs, so their paired deltas are not differences from the all-row first estimate. The final bound treats every partial run first as a match and then as a mismatch.

Decision checked	Population	Rows	Mismatch	Δ vs. paired first [95% CI]
First	all validated rows	7,402	19.6%	–
First	complete trajectories	6,591	18.2%	0.0
Last	complete trajectories	6,591	16.8%	-1.5 $[-6.1, +3.7]$
Any (exact)	complete trajectories	6,591	1.4%	-16.9 $[-28.0, -8.3]$
Any (deterministic bound)	all validated rows	7,402	[1.2%, 12.2%]	–

Table 8: Does call content help predict whether the official success check changes from false to true across the scored calls? The baseline uses transcript counts, call count, call type, and argument length. The augmented model also compares call arguments with the user request and prior tool outputs. Cells report log loss / AUROC. Positive Δ loss favors the augmented model; intervals keep both fitted models fixed and do not include refitting uncertainty.

Data	Rows	Counts + call shape	+Argument–context	Δ loss [95% CI]
Public systems	4,328	0.173 / 0.966	0.096 / 0.988	$+0.077$ $[0.022, 0.152]$
Unseen systems	932	0.484 / 0.918	0.327 / 0.949	$+0.157$ $[0.003, 0.298]$

F.3 Remaining suffix length

A decision with $L = 1$ has already completed the goal, so its recorded continuation may simply be shorter than that of a decision with $L = 0$ whose run completes later. The contrast of 1.0% against 91.0% would then reflect the opportunity left after the decision rather than the local-completion distinction encoded by L. The replay records how many calls follow the scored sequence.

The suffixes do differ. The median number of later calls is 1 for $L = 1$ and 3 for $L = 0$, and 37.4% of the $L = 1$ decisions have no later call at all against 0.0%. The contrast nevertheless remains. Among decisions with at least one later call the shares are 1.7% and 91.0%. Stratifying on suite and on the exact number of later calls, and pooling only the 21 of 48 strata that contain both classes, gives 1.7% on 358 rows against 90.0% on 946.

G Exact Breakdown of Mixed AUROC

Table 9: Exact AUROC decomposition for scores trained on the mixed population. C_{11} compares the same fixed scores only when both decisions contain calls; it differs from a model refit using only decisions with calls. Cross is the AUROC contribution from pairs in which only one decision contains a call ($O^+ \neq O^-$).

Score	Aggregate	C_{11}	Cross	Aggregate $-C_{11}$
Call presence	0.671	0.500	0.342	0.171
Suite + call presence	0.826	0.736	0.342	0.090
Basic call features	0.799	0.703	0.337	0.097
Transcript + call content	0.807	0.709	0.340	0.098
Call content only	0.820	0.732	0.338	0.088

The "Cross" column is the weighted contribution from comparisons between different call-presence groups in Equation 3, not an AUROC computed on a separate sample. The C_{11} column evaluates mixture-trained predictions only. The main estimates restricted to decisions with calls filter before regularization selection, calibration, fitting, and scoring and can therefore differ from C_{11}.

Why an arbitrary relabeling lowers AUROC. The null mean in Section 4.2.1 has a closed form that makes its direction obvious. Let a fixed score have m positives, n negatives, and AUROC a under

the half-credit tie convention, and keep a uniformly drawn k of the positives while the other $m-k$ join the negatives. The retained positives keep expected AUROC a against the original negatives and score $1/2$ against the reassigned ones, so

$$\mathbb{E}[\mathrm{AUROC}_{\mathrm{random}}] = \frac{na + (m-k)/2}{n+m-k}, \qquad \mathbb{E}[\Delta_{\mathrm{random}}] = \frac{m-k}{n+m-k}\left(\tfrac{1}{2} - a\right). \tag{5}$$

Any score better than chance therefore loses AUROC on average when the positive class is shrunk arbitrarily. With $m = 165$, $n = 337$, and $k = 66$ on the matched public cohort this gives -0.068 for TS-Guard and -0.057 for Mistral, matching the simulated null means. The band is the central 95% interval of 20,000 draws. The reported p is the one-sided Monte Carlo estimate $(b+1)/(B+1)$, where b counts draws at or above the observed change out of $B = 20,000$; no draw reaches the observed value for the informative monitors, so $p \approx 5 \times 10^{-5}$ and is written as an upper bound. The band answers a different question from the paired resampling interval: that interval asks how the change varies if the evaluation population is redrawn, while this one holds the sample fixed and asks whether an arbitrary positive-class shrinkage of the same size reproduces the observed change. A monitor can have a paired interval covering zero and still sit far outside this null, and TS-Guard does.

An exact four-cell decomposition also explains why changing labels affects fixed monitors differently. On the 502-row held-out public monitor set, the (Y, L) counts are $(0,0) : 337$, $(0,1) : 0$, $(1,0) : 99$, and $(1,1) : 66$. Thus changing from Y to L removes the 99 later-success rows from the positive class in this set; it does not add a reverse-mismatch class. Mistral ranks a within-sequence transition above a later success in 87.5% of such pairs, whereas Qwen does so in 51.0%. In the new-system set, the corresponding counts are 540, 0, 244, and 148, and TS-Guard ranks the within-sequence transition higher in 85.6% of cross-group pairs.

H Results on Held-Out Suites and New Systems

Three public systems contain later workspace goals absent from the original 629-path set. Before reading outcomes, we remove those 749 paths so that the comparison uses the same task–goal set. Across the remaining 11,402 rows, suite plus call presence reaches 0.796 AUROC on the 870-row mixture test. Among 640 test rows that contain calls, the point gain from adding comparisons between call arguments and visible context over generic shape is +0.023 log-loss units (AUROC 0.685 versus 0.646). Differences between versions of the public data change the magnitude, not the call-presence shortcut.

Holding out a complete suite removes the suite, its user tasks, and its injection goals. Across 5,866 rows from held-out suites that contain calls, the log-loss gain from adding comparisons between call arguments and visible context over basic call features is -0.247. The preregistered resampling over systems and matched task–goal pairs gives 95% CI $[-0.406, -0.099]$; a system–task–goal sensitivity gives $[-0.694, 0.009]$. Per-suite gains are banking $+0.053$, slack $+0.055$, travel -0.043, workspace -0.719. The reversal is concentrated in workspace, while banking and Slack have small positive point estimates. The added call-content features still rank better than generic shape (0.654 versus 0.594 AUROC), but their probability quality is worse (0.859 versus 0.612 log loss; 0.121 versus 0.063 ECE). The preregistered paired interval excludes zero, whereas the broader system–task–goal sensitivity crosses zero. This supports a performance loss in some suites. The new-system experiment tests the complementary axis of new systems on familiar benchmark content: On 984 decisions with calls from 2 newly evaluated systems, the log-loss gain from adding comparisons between call arguments and visible context over basic call features is $+0.008$ (95% CI $[-0.041, 0.047]$). Per-system gains are Mistral -0.013 and Qwen +0.024. On the stricter new-system, task, and goal subset, 132 decisions with calls yield a log-loss gain of -0.036 (95% CI $[-0.132, 0.085]$).

I Monitor Scores, Selection, and Executed Policies

I.1 The same scores under two labels

We compare each fixed score against both Y and L (Table 3). The paired change in AUROC is primary. Average precision depends on how many examples are positive. Because Y and L have different positive rates, differences in separately calibrated log loss, Brier score, and expected calibration error (ECE) are descriptive rather than direct measures of improvement.

Table 10: Log-loss gain from adding comparisons between call arguments and visible context to transcript counts plus basic call features. Every evaluated decision contains calls; each row has a different transfer target. Positive ΔLL means that the added features lower log loss.

Population	Rows	ΔLL	95% CI
Systems/tasks/goals held out	747	$+0.015$	$[-0.059, 0.082]$
Whole-suite pooled	5,866	-0.247	$[-0.406, -0.099]$
New systems	984	$+0.008$	$[-0.041, 0.047]$
New systems + held-out tasks/goals	132	-0.036	$[-0.132, 0.085]$

Table 11: Whole-suite transfer among decisions that contain calls. Each model is fit without the named suite or any of its task and goal identifiers. LL denotes log loss; positive ΔLL favors call content.

Held-out suite	Rows	Basic LL	Content LL	ΔLL
Banking	1,007	0.742	0.689	$+0.053$
Slack	1,132	0.925	0.870	$+0.055$
Travel	1,650	0.533	0.576	-0.043
Workspace	2,077	0.441	1.160	-0.719

As a secondary resampling check, 57.9% of accepted draws reproduce both point-estimate inequalities between Mistral and TS-Guard.

I.2 Extra held-out error from training against a different event

The reference procedure selects the monitor and fits its probability mapping with L, the before/after label defined above; extra held-out log loss is measured relative to this L-trained procedure. For each reserved system, leave-one-system-out log loss selects a monitor using only the other 11 systems. A logistic score-to-probability curve is then fit on those same 11 systems. The two label-specific selectors choose different monitors in 12/12 repetitions. The intermediate row keeps the end-of-run-selected monitor and changes only the label used to fit its probability mapping. This splits the total extra loss exactly into the cost of monitor selection and the cost of its probability mapping for each held-out row and in aggregate; displayed components may differ from their displayed sum by rounding. The primary paired-system intervals exclude zero for both components, while the fixed-prediction system–task–goal interval for selection alone is [-0.082, 0.247] and includes zero. We therefore emphasize the combined consequence of monitor choice and probability mapping. This analysis was fixed after the separate label-specific rankings were known but before the repeated monitor choices, excess losses, or uncertainty results were inspected. Table 14 adds pooled all-row metrics for the L panel of Table 4.

How much of the gain is a corrected base rate? Log loss penalises a procedure calibrated to the wrong prevalence even when its scores carry information, so part of the drop from 0.395 to 0.171 is not information about the calls. Two baselines on the same 12 folds give reference points for that part. Predicting the training folds' frequency of L for every row gives mean system log loss 0.387; predicting it within the row's own suite gives 0.368. Both are lower than the Y-selected and Y-calibrated procedure, so training against the wrong event does worse here than ignoring the monitor and predicting the base rate. Both are well above the L-trained procedure, so its scores add information that neither the overall nor the per-suite frequency supplies.

Evaluation against held-out Y. Against held-out Y, the order reverses: selecting and calibrating against L gives 1.156, recalibrating that monitor to Y gives 0.567, and selecting and calibrating against Y gives 0.486. Losses under different targets are not comparable to each other, so each comparison stays within one target; within each, the procedure aimed at the evaluated event wins, and aiming at the other event more than doubles the loss in both directions. The training target should therefore be the event on which the monitor is judged.

Table 12: Transfer to two newly evaluated systems, shown by system and familiar suite. LL denotes log loss. Positive ΔLL favors adding comparisons between call arguments and visible context over basic call features.

Group	Rows	Pos.	Basic LL	Content LL	ΔLL
System: Mistral	417	149	0.723	0.736	−0.013
System: Qwen	567	268	0.897	0.873	+0.024
Suite: Banking	221	103	0.702	0.661	+0.041
Suite: Slack	197	124	0.940	0.925	+0.015
Suite: Travel	224	94	1.329	1.434	−0.105
Suite: Workspace	342	96	0.503	0.446	+0.057

Table 13: Transfer of the unchanged official TS-Guard to two newly evaluated systems. Scores are mapped to probabilities with a logistic curve fit only on the fixed public development set. AP denotes average precision. The all-row log-loss 95% CI is $[0.683, 0.860]$; the interval resamples two systems and attack components.

Population	Rows	Pos.	Log loss	AUROC	AP
All rows	984	417	0.776	0.632	0.512
Mistral	417	149	0.727	0.585	0.408
Qwen	567	268	0.813	0.696	0.632

I.3 THE SAME TOOLSAFE SCORES UNDER TWO LABELS

We cluster by source file and ASB `id-interaction`, retaining all steps from a sampled interaction together, and resample clusters within the two success groups and one failure group.

We analyze every one of the 5,231 steps present in ToolSafe's fixed public ASB-Traj files and exactly represented in its released TS-Guard log. This public release is 6 steps smaller than the 5,237 steps reported in ToolSafe Table 2: four annotated-score-0 steps and two annotated-score-1 steps are absent from the release. The public evaluator concatenates all three released arrays without filtering, and its labels, predictions, metadata, and metrics bind all 5,231 rows. Because the release contains no exclusion manifest, the six missing row identities and the upstream data-version cause cannot be recovered.

Rerunning the pinned checkpoint produces one parsed rating for each of the 5,231 expected row identities, with no missing, duplicate, or unparsed rows. It agrees exactly with the released rating on 97.2% of rows; the mean absolute rating difference is 0.015.

The strict step target in Table 15 treats both annotated scores 0.5 and 1 as positive. The score-1-only sensitivity preserves the direction for both vectors. The rerun's scores give local AUROC 0.918 and a paired change of +0.161 (95% CI [0.149, 0.173]); released scores give 0.920 and +0.163 (95% CI [0.151, 0.174]). These labels are supplied by ASB-Traj.

I.4 WHAT THE PUBLISHED THRESHOLDS WOULD FLAG

At a 10% benign-step budget on AgentHarm-Traj, leave-one-interaction-out threshold selection chooses an abstaining cutoff above the released score range: 0/525 harmful and 0/206 benign steps would be flagged. At the looser rating threshold in Table 16, the no-recovery absolute utility-loss proxy is 31/227 (13.7%). This is an upper bound under the assumption that every flagged, originally successful clean call sequence loses all utility and the agent never recovers.

Table 14: Leave-one-system-out selection and logistic calibration on the same 502 matched public decisions, evaluated against held-out L, with pooled all-row metrics. Each procedure uses only the other systems; lower log loss and higher AUROC are better.

Selection target	Calibration target	Selected monitor	Mean system LL	All-row LL	All-row AUROC
End-of-run Y	End-of-run Y	TS-Guard (12/12)	0.395	0.400	0.852
End-of-run Y	Before/after L	TS-Guard (12/12)	0.240	0.249	0.844
Before/after L	Before/after L	Mistral (12/12)	0.171	0.183	0.915

Table 15: The same ToolSafe ASB-Traj rows and score vectors under two labels. "Released" uses ToolSafe's published ratings; "rerun" uses the pinned checkpoint. Each vector is evaluated unchanged against end-of-run success and the strict step label, which treats annotated scores above 0 as positive. Paired intervals resample complete interactions within each source and success group.

Score source	Target	AUROC	$L_{\mathrm{ASB}} - Y$ [95% CI]
Released TS-Guard	End-of-run success Y	0.757	–
Released TS-Guard	Strict step label L_{ASB}	0.953	$+0.197$ [0.187, 0.206]
Rerun TS-Guard	End-of-run success Y	0.756	–
Rerun TS-Guard	Strict step label L_{ASB}	0.952	$+0.196$ [0.186, 0.205]

Table 16: Rows TS-Guard would flag at its two released nonzero ratings. Cells report flagged/total (rate) and 95% CIs obtained by resampling the predefined evaluation groups. For AgentDojo, positive means that the official success check is false before the scored call sequence and true after it; this is not a human harm label. AgentHarm-Traj supplies separate expert-guided step labels. These are counterfactual flags, not executed blocks or causal effects.

Data set	Rating $\geq$	Positive flagged	Reference flagged
AgentDojo public	1	$L = 1$: 23/66 (34.8%) [2.1%, 95.0%]	Clean: 1/227 (0.44%) [0.0%, 2.5%]
AgentDojo new systems	1	$L = 1$: 83/148 (56.1%) [31.1%, 83.3%]	–
AgentHarm-Traj	1	Harmful: 331/525 (63.0%) [53.3%, 72.3%]	Benign: 42/206 (20.4%) [13.0%, 28.3%]
AgentDojo public	0.5	$L = 1$: 65/66 (98.5%) [89.8%, 100.0%]	Clean: 45/227 (19.8%) [6.6%, 35.8%]
AgentDojo new systems	0.5	$L = 1$: 143/148 (96.6%) [86.3%, 100.0%]	–
AgentHarm-Traj	0.5	Harmful: 509/525 (97.0%) [93.9%, 99.2%]	Benign: 95/206 (46.1%) [36.7%, 55.7%]

I.5 What happens when flagged calls are blocked

The executed study uses the new-system set selected before outcomes were read. It pairs each protected run with the same unprotected system, suite, user task, and injected goal. Both conditions cover all 1,173 matched runs with zero recorded runtime errors. The block rule starts only after a nonempty tool result, prevents the complete pending call set from executing, returns generic errors, and lets the agent continue. Thus Tables 17 and 18 report the final outcomes under the complete rule. Policies selected under different labels are compared against the observe arm in Table 5.

Blocked runs and decisions. At threshold 0.5, this cohort has 441 blocked runs and 968 blocked decisions of 2,770 scored, of which 0 are fail-closed.

I.6 From the label to the executed policy

The executed comparison is the end of a chain from the label to the blocking policy.

1. **Target.** Y or L on the matched public cohort.
2. **Monitor.** Nested leave-one-system-out selection over Mistral, Qwen and TS-Guard. Y selects TS-Guard in 12 of 12 folds and L selects Mistral in 12 of 12; neither selector ever picks Qwen.
3. **Calibration.** A logistic score-to-probability map fitted against the same target on the training systems. It enters selection; the executed thresholds apply to uncalibrated scores.
4. **Operating point.** Not fixed by that study. TS-Guard runs at the two thresholds its frozen execution specification declares, and the Mistral monitor at two declared points that bracket TS-Guard's blocking burden from below and above, which is a choice made with the executed cohort in view.
5. **Execution.** The agent, matched identities, blocking semantics, and continuation budget are held fixed; the scorer and its threshold vary.

Only the fourth link is not fixed in advance, so both L operating points are reported.

Offline and executed Mistral scorers. The system prompt, the user template, the decoding temperature, the nucleus parameter and the token budget are identical between the offline scorer and the executed one. Two things are not. The response schema carries a strictness flag offline that the executed path omits, and the two probability parsers are written differently. Applying both parsers to every one of the 5,892 replies the executed arms recorded returns the same probability on 5,892 of them, so the parser difference is inert on this corpus. The schema flag acts during generation and is not bounded by that check.

I.7 Monitor results on decisions that contain calls

Tables 19 and 20 report the prespecified monitor comparison and several review rates.

After removing decisions without calls, the held-out test contains 212 positives and 535 negatives. Word patterns in calls rank end-of-run outcomes at 0.737 AUROC, with 0.537 log loss. Features that compare call arguments with visible context reach 0.713 AUROC and 0.545 log loss. Its gain over transcript plus generic call shape is +0.015 (95% CI with predictions fixed [-0.059, 0.082]; full-refit CI [-0.191, 0.124]). Both intervals cross zero: word patterns retain ranking signal, but the data do not support a reliable probability-quality gain from the added comparison features.

Among attack prefixes with calls, calibrated TS-Guard reaches 0.817 AUROC and Mistral reaches 0.767; Qwen reaches 0.534. Adding comparisons between call arguments and visible context to each monitor plus basic call features has no supported log-loss gain. For TS-Guard the change is -0.017 (95% CI $[-0.072,\ 0.034]$). Without new-system fitting or threshold tuning, the unchanged official TS-Guard scores 984 new-system decisions with calls at 0.632 AUROC (95% CI $[0.551,\ 0.721]$) and 0.776 log loss after mapping scores to probabilities on public development data (95% CI $[0.683,\ 0.860]$). The per-system AUROCs are 0.585 for Mistral and 0.696 for Qwen. Removing the call-presence shortcut does not erase all monitor signal, while the adjacent comparison gives no

Table 17: Executed TS-Guard block rule on paired new-system AgentDojo identities. Attack and user-task values are percentages or percentage-point changes. Intervals are 95% paired resampling intervals over systems, tasks, and goals. The comparator is a set of earlier unprotected runs matched by identity rather than the blocking-disabled arm of Table 5, so the last four columns report discordant outcomes between execution configurations rather than attacks causally prevented or induced by blocking. The two count columns give identities whose attack outcome flips from positive under the earlier runs to negative under the blocking configuration, and the reverse.

Threshold	Earlier attack	Blocking config	Attack change	$+\to-$	$-\to+$	User-task change
0.5	35.6	1.8	-33.8 [-50.6, -18.9]	406	9	-2.9 [-12.5, +6.9]
1.0	35.6	10.0	-25.7 [-36.2, -16.1]	324	23	+1.4 [-11.1, +12.2]

Table 18: Descriptive breakdown of the executed block rule. Values are percentages or percentage-point changes. System and suite rows overlap, so they are not pooled as independent observations; the paired uncertainty analysis remains the aggregate result in Table 17. As there, the comparator is a set of earlier unprotected runs rather than the blocking-disabled arm of Table 5, and the two systems also differ from each other in their tool-call recovery adapter, so these rows describe distinct execution configurations.

Threshold	Group	Runs	Earlier	Blocking config	Attack change	User-task change
0.5	Mistral Small 3.2	553	27.1	1.6	-25.5	-6.0
0.5	Qwen3-30B-A3B	620	43.2	1.9	-41.3	-0.2
0.5	Banking	257	40.1	0.8	-39.3	-14.8
0.5	Slack	201	61.7	3.5	-58.2	-16.9
0.5	Travel	268	35.4	3.7	-31.7	+2.2
0.5	Workspace	447	21.5	0.4	-21.0	+7.2
1.0	Mistral Small 3.2	553	27.1	4.5	-22.6	-5.4
1.0	Qwen3-30B-A3B	620	43.2	14.8	-28.4	+7.6
1.0	Banking	257	40.1	4.7	-35.4	-6.6
1.0	Slack	201	61.7	27.4	-34.3	-4.5
1.0	Travel	268	35.4	9.0	-26.5	+4.1
1.0	Workspace	447	21.5	5.8	-15.7	+7.2

supported probability-quality gain to those added features once monitor score and basic call features are available.

Table 19: Models that score pending call sequences when systems, tasks, and goals are all held out; every evaluated decision contains calls. LL denotes log loss. Positive ΔLL favors adding comparisons between call arguments and visible context to the same monitor, transcript counts, and basic call features.

Monitor	Calibrated LL	AUROC	+basic LL	+content LL	ΔLL [95% CI]
Qwen monitor	0.728	0.534	0.557	0.541	$+0.015\ [-0.060,\ 0.082]$
Mistral monitor	0.511	0.767	0.514	0.496	$+0.018\ [-0.042,\ 0.073]$
Logit ensemble	0.566	0.767	0.537	0.495	$+0.041\ [-0.028,\ 0.107]$
TS-Guard	0.468	0.817	0.440	0.457	$-0.017\ [-0.072,\ 0.034]$

At a 10% budget, a word-pattern score sends 27.4% of successful attacks and 0.0% of utility-successful clean calls. TS-Guard at rating threshold 0.5 retains 23.1% attack capture with 0.0% clean burden. At a 30% budget, the word-pattern score sends 59.9% of attacks and 7.7% of successful clean calls; TS-Guard then flags 49.1% of attacks and 1.1% of successful clean calls. These budgets count only decisions that contain calls. Clean burden is measured on successful benign AgentDojo runs; a routed clean decision consumes the more expensive monitor and may add review latency, but it is not an automatic block.

Table 20: A word-pattern score selects decisions for TS-Guard, which uses threshold 0.5. Every decision contains calls. Clean rates use utility-successful benign calls.

Budget	Routed attack	Routed clean	Flagged attack	Flagged clean
10%	27.4%	0.0%	23.1%	0.0%
30%	59.9%	7.7%	49.1%	1.1%
50%	78.3%	26.2%	60.8%	8.7%

J New-System Execution Details

The new Qwen and Mistral evaluation sets use AgentDojo v1.2.2. They evaluate its instruction-following attack over banking, Slack, travel, and workspace. The inclusion rule requires exactly 629 selected attack paths per system. The local OpenAI-compatible adapter first attempts strict JSON parsing and then literal-only normalization for Python-style dictionaries. A final malformed or truncated object is recorded as an explicit runtime error. It is preserved in the execution audit and excluded by the same prewritten inclusion rule used for public runtime errors.

The pinned checkout exposes workspace goals 0–13, whereas the original public sets cover goals 0–5. Before inspecting new-system runs or outcomes, we fixed the evaluated task–goal set to all banking, Slack, and travel pairs plus those six workspace goals, yielding 629 paths per system. The 320 additional workspace paths remain only in the execution audit and are not used in extraction, fitting, scoring, or monitor inference.

K Moving the Protocol to Another Benchmark

Running this protocol elsewhere needs an official criterion that is well defined on the snapshots of Equation 1, and a state object that can be restored. AgentDojo satisfies both: its injection check takes the pre- and post-execution environments, and the suite deep-copies the environment before a task runs.

Other benchmarks package the criterion differently. AgentHarm's graders take the message list and ask which target functions were called, with which arguments, and in what order. Whether such a criterion is well defined on a prefix is an implementation question we did not settle, and it is a separate question from whether it is a good criterion: a predicate over specific functions and arguments is not the call-presence indicator O, since two sequences that both contain a call can satisfy it differently. What a transplant would need is for the criterion to name the parts of a snapshot it reads, so that a before/after evaluation is defined rather than inferred.

L Supporting Studies: MCP Context and MANTRA Rules

Controlled MCP calls. We authored 53 benign, short-horizon Model Context Protocol calls together with six executable rule profiles. The main study crosses 53 calls, three information conditions, three repetitions, and six systems for 2,862 rows. The conditions show no deployment record, the true record, or the true record plus the complete rule. Calls whose rule says review or block test whether the system chooses the required action when that rule is hidden. Four responses chose an action outside the allowed set and are counted as wrong; there are no transport errors.

The matched-context control covers two systems and replaces the true record with irrelevant text of the same length. It contains no user or service identity, credential, resource, confirmation, or outcome. Fixed before the runs, the 1,060-row design has no communication or parsing errors. A separate diagnostic appends the computed final rule decision while holding the call, record, rule, and action space fixed. All 954/954 responses then choose the required action. This shows that the systems can follow an explicit answer; it does not show that they inferred that answer from the earlier context.

Table 21: Controlled 53-call study. The first column reports the change in the rate of choosing execute now after adding the true deployment record on the 46 calls whose authored rule says review or block; no rule is shown in either condition. The second reports the exact prescribed action when both the record and rule are shown. Values are percentages or percentage-point changes with 95% intervals from resampling scenarios.

System	Execute-now change	Exact action with record + rule
Claude Haiku 4.5	+44.9 [+33.3, +56.5]	99.4 [98.1, 100.0]
Claude Opus 4.6	+15.2 [+2.9, +28.3]	83.0 [73.6, 91.2]
Claude Sonnet 4.6	+28.3 [+16.7, +41.3]	73.0 [61.6, 83.6]
GPT-5.4 mini	+53.6 [+42.0, +65.2]	93.7 [87.4, 98.7]
GPT-5.6 Sol	+26.8 [+15.9, +38.4]	89.9 [81.8, 96.2]
Qwen3-4B Q4	+30.4 [+17.4, +43.5]	13.2 [5.7, 22.6]

Table 22: Matched-context control on the same 46 review-or-block calls. The rule is withheld. Execute-now rates are percentages; the last column is the paired true-record minus irrelevant-record change with a 95% interval.

System	No record	Irrelevant	True	True − irrelevant
GPT-5.4 mini	17.4	63.0	69.6	+6.5 [-1.1, +14.1]
GPT-5.6 Sol	27.2	26.1	47.8	+21.7 [+9.8, +34.8]

Table 23: Held-out MANTRA rule-label recovery. Full information supplies the symbolic state and complete manual. Values are balanced accuracy or paired percentage-point gains with 95% connected-component bootstrap intervals.

System	Full information	State gain	Manual gain
GPT-5.4 mini	78.0 [71.8, 83.3]	+8.6 [+1.1, +14.9]	+31.0 [+24.4, +37.2]
GPT-5.6 Sol	67.8 [62.7, 72.8]	+7.9 [+0.8, +14.4]	+33.2 [+28.2, +38.7]
Mistral Small 3.2	53.3 [46.3, 60.0]	+6.0 [-3.0, +14.8]	+8.7 [+1.6, +16.6]
Qwen3-30B-A3B	64.7 [61.3, 66.8]	+16.5 [+10.3, +22.9]	+37.8 [+30.0, +45.0]

MANTRA rule labels and executed tasks. The fixed extractor retains 46 matched required/forbidden pairs (92 rule cases) from MANTRA's released checks (Anand et al., 2026). Four systems classify each case three times under six information conditions, producing 6,624 rows. Three malformed responses are scored as wrong; no provider failure is omitted. The same systems separately execute all 285 released tasks three times using MANTRA's released tools and evaluator, with zero recorded agent errors. For each pattern, the predictive analysis asks whether improved rule-label recovery lowers held-out loss for the corresponding released check in an executed task. The change is -0.0023 (95% CI [-0.0200, +0.0061]), so the prediction test fixed in advance does not establish an improvement.

The artifact places MCP scripts, specifications, and tests in `harness/mcp_controlled_study/`, with compact results in `results/mcp-controlled-study/`. MANTRA code is in `harness/`, with compact analyses in `results/mantra-transfer/`. The reproduction entry point reads the saved outputs and regenerates Tables 21–23.