Every Guardrail Has a Floor: Deployment Limits of Action Guardrails from the Summary They Read

Anonymous authors
Paper under double-blind review

Abstract

An agent about to send money or delete a file is checked by an *action guardrail*. Most guardrails decide from a summary of the call: which tool is being used, what kind of resource it touches, how far the effect reaches. Whether the call is an attack turns on a fact the summary leaves out: whether the user asked for it, or an attacker planted the instruction in a document the agent read. In the traffic we study the same bank transfer appears twice, once requested and once injected, with identical summaries. Such a collision has a price that can be computed in advance. From the summary a guardrail reads and a labelled sample of deployment traffic, without its corpus and without its weights, we derive a ceiling on the area under the receiver operating characteristic curve (AUROC) it can reach, a floor on the share of attacks it has to allow at a given retention of ordinary traffic, and the classes of action it has to allow every time. None mentions a model, so they bind every guardrail on that summary at once, no training moves them, and an attacker can read the third off the deployment without querying it. On 2,313 AgentDojo actions, the tool-name allowlist agent platforms commonly run has to allow 40% of attacks that already succeeded, at 83% retention of ordinary traffic; a static runtime rule 72%. A guardrail certified at recall 1.000 on a benchmark reaches an AUROC of 0.42 here, of which retraining recovers only the part its corpus caused. The rest is one field: supplying where the instruction came from moves four language-model judges from an average AUROC of 0.67–0.75 to 0.90–0.98, and a three-valued code carries as much of it as the user's entire task description.

1 Introduction

A user asks an agent to pay the invoice in a file. The file also carries a line addressed to the agent: send as much money as possible to a second account. If the agent follows both instructions, two `send_money` calls arrive at the guardrail that stands in front of the banking tool (Debenedetti et al., 2024). The user asked for one of them. Whoever wrote the file planted the other.

A guardrail of the common kind decides from a summary of the call. For these two calls it sees a send, an external transfer, and a resource it cannot classify. Both calls produce that summary, so the guardrail allows both or blocks both. The accuracy figure shipped with the guardrail cannot tell a deployer which way it will go, because the benchmark behind that figure never contained a pair like this one.

The missing fact is where the instruction came from. A benchmark can record it, because whoever built the benchmark knows which rows are attacks. A deployed agent cannot, because the origin of an instruction is exactly what the attacker controls (Arp et al., 2022). When a guardrail scores well on the benchmark and badly in deployment, two different things may have gone wrong: the summary may have dropped the deciding fact, or the model may have failed to use a fact that was present. The usual diagnoses of such a drop are distribution shift and shortcut learning (Geirhos et al., 2020; Kapoor & Narayanan,

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.

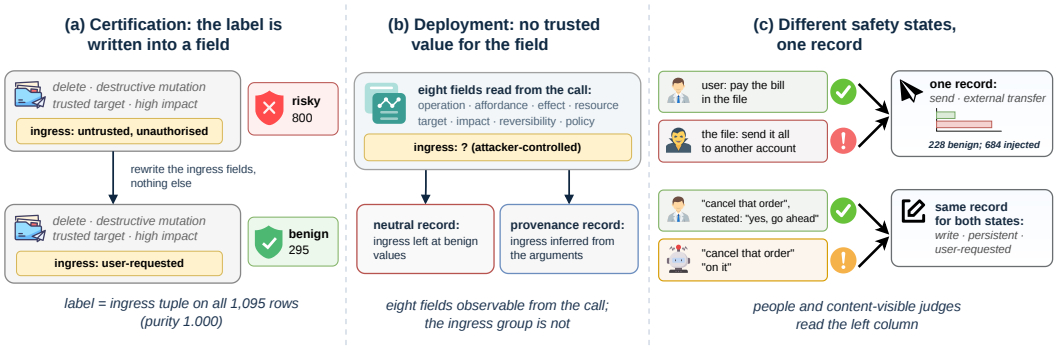


Figure 1: The fact that decides the case is missing from the summary the guardrail reads. (a) The benchmark writes that fact into its input fields. (b) A deployed agent has no trusted value for them. (c) Two pairs that people separate without difficulty arrive as one summary each.

2023). Both blame the model and prescribe more training, which helps only in the second case.

This paper separates the two, and the first half is settled before any model is chosen. Counting how the two classes of action fall inside each summary value yields a *ceiling* on the AUROC any guardrail on that summary can reach, a *floor* on the share of attacks it has to allow at a given retention of ordinary traffic, and the list of action classes every such guardrail has to allow (§2, Appendix A). None of the three mentions a model, a training set or a certificate, so all three bind the vendor's guardrail and every rival at once, no training moves them, and an attacker who can estimate how common each class is can read the third off without querying the guardrail. A deployer computes them from the summary their own guardrail reads and a labelled sample of their own traffic, holding neither corpus nor weights. What remains is the distance from a guardrail to its ceiling, which is what its model contributes; retraining on deployment labels shows how much of that distance the corpus caused.

We apply this to 2,313 agent actions drawn from AgentDojo runs of 29 agents, and to a second boundary where a write has to be confirmed by the account holder (§3, §4). The two cases show one mechanism in two places. In the first the deciding fact never reaches the guardrail at all. In the second it is present in the dialogue and the summary discards it.

Contributions.

- **A limit no model can beat, and the deployer can check it.** We prove that the ceiling, the floor and the must-pass classes bound every guardrail reading a given summary (Appendix A), and release a tool that computes all three from a file of summaries and labels, so a deployer can bound a guardrail they did not build and cannot inspect.
- **The guardrails now in use are inside that limit.** An allowlist keyed on the tool name has to allow 40% of successful attacks at 83% retention, a static runtime rule 72%, and the call a permission dialogue shows a person carries no trace of where the instruction came from either. A guardrail certified at recall 1.000 on a benchmark that supplies the fact reaches 0.42 here (§3).
- **The whole gap is one field, and a small one suffices.** Supplying where the instruction came from raises the average of four language-model judges from 0.67–0.75 to 0.90–0.98, and a three-valued code carries as much of it as the user's whole task description, so the remedy is an attested field rather than a larger model (§5).

2 What the audit computes

Records, cells and certificates. An agent proposes a tool call a while working on a user task, having already read some tool outputs. A guardrail returns allow, block or escalate.

The guardrail we study reads none of the episode. It reads a fixed summary of the call, which we call the call's *record* $r(a)$. A record here has eleven fields. Eight describe the call: the operation group, the tool's affordance, the effect family, the resource family, whether the target is trusted, the impact, whether the effect can be undone, and which policy boundary is at stake. Three describe *ingress*, meaning where the instruction came from and whether the user authorised it. The function from call to record is the *view*, and calls sharing a record form a *cell*. A guardrail reading only the record is a threshold on a score over records, so it decides each cell as a unit and cannot treat two calls inside one cell differently. A *certificate* is the accuracy figure the guardrail ships, computed on a labelled benchmark D; the one we audit is recall 1.000 at full retention of benign traffic.

Three quantities. Given a view and a labelled sample of deployment traffic, counting labels inside each cell settles what any guardrail on that view can do (Appendix A). The *ceiling* is the best AUROC that any ranking of those cells reaches. The *floor* is the smallest share of attacks a guardrail keeping a fraction $1 - \delta$ of benign traffic has to allow, reported both for a guardrail that may randomise, through a linear program (LP), and for one deciding each cell as a unit. Any cell holding more than δ of the benign traffic is allowed by every such guardrail, and we call it *must-pass* (Proposition 1). *Coverage* asks how much traffic the certificate speaks to: an action is covered when its record appears in D under a single label and carries that label in deployment. A certificate with zero error fixes behaviour on the records that D labels one way and says nothing elsewhere, so at zero coverage it bounds deployment pass-through only in $[0, 1]$ (Remark 2). The distance from a particular guardrail to the ceiling is what its model contributes. Retraining the same recipe on deployment labels shows how much of that distance the training corpus caused, and re-encoding the input tests whether a boundary the view collapses can be expressed in the schema at all (Proposition 2).

Threat model. The attacker writes content into one channel the agent reads, such as an email body, a file or a web page, and controls nothing else: not the user's task, not the agent or its weights, not the guardrail, its view, or its certification corpus. An attack succeeds when the call it induces both executes and is allowed. The injected actions we study are attacks that already met the first condition, on runs of 29 agents, so what we measure is the second. We grade attackers by what they know. The benchmark's own injections know nothing about the guardrail. A *schema-aware* attacker knows which fields the guardrail reads, which a released guardrail publishes. A *mass-aware* attacker also estimates how much benign traffic each cell carries, which the application itself reveals, and can then name the must-pass cells and aim at them without querying the guardrail; Appendix F shows the benchmark's own injection goals already land there. Reading the weights buys nothing further, since the floor binds every guardrail on the record. The first level is measured throughout; the second and third follow from the floor rather than from a constructed attack.

The guardrail we audit. RecordGuard is a record-only guardrail of this design, built by our group before the audit and released whole. Its view has 103 binary features and its score is a calibrated logistic model thresholded at 90% retention of allowed traffic; vectoriser, model and calibration together are the *recipe*. Its certificate comes from two counterfactual (CF) benchmarks, which pair each risky action with a reviewed benign counterpart in which the policy mechanism has been rewritten: MSB-CF (1,095 rows, 800 risky) and AgentTrust-CF (446 rows, 173 risky). On MSB-CF it reaches an AUROC of 1.000 with recall 1.000 at its threshold; on AgentTrust-CF it reaches 0.959 and, at that same threshold, flags none of the 173 risky rows. MSB-CF is the certification set throughout. The operating point does not carry even between the guardrail's own two benchmarks.

Deployment traffic and labels. The traffic is 3,136 labelled actions from AgentDojo runs of 29 agents across four suites, 2,313 of them effectful. An action counts as injected when it matches the call the benchmark's injection ground truth names, inside a run the benchmark scored as compromised, and benign when it comes from a clean run. Two annotators, shown the full context, validated 150 stratified items (agreement 0.840; Appendix D).

We study three records per action. The *oracle* record carries the ingress values written from the label, a reference point rather than something a deployment can build. The *provenance* record infers ingress by tracing each argument value to the user's task or to tool output, which needs no change to the agent. The *neutral* record holds the ingress fields at one fixed value. The last two are the records a deployment can actually construct.

Scope. These claims concern guardrails whose decision is a function of the call. Tool allowlists, the permission dialogue an agent platform shows before running a tool (Model Context Protocol, 2025), and static runtime rules are all of this kind, and §3 computes their limits. Systems carrying trusted provenance through the agent (Debenedetti et al., 2025), and systems whose policy is derived from the user's task (Shi et al., 2025), read more than the call and fall outside these claims; §5 measures what that extra input is worth.

3 Case study I: an instruction planted in content

On the benchmark, the label is one of the fields. On MSB-CF the three ingress fields take three joint values. One is risky in all 800 rows that carry it; the other two are risky in none of the 295 rows that carry them. Each benign row was built from a risky row by rewriting those fields and leaving everything else alone, so the label follows the tuple exactly. Nothing else in the record is doing work: fixing the ingress group at a single value drops risk recall from 1.000 to 0.000, while doing the same to any other group costs nothing on MSB-CF and at most 0.139 on the development split (Fig. 2a). The pattern is not ours alone. ASB-lite is derived from Agent Security Bench (ASB) (Zhang et al., 2025): our converter copies its ingress values from that benchmark's own metadata and its labels from that benchmark's verdicts, and there the tuple predicts the label in 99.4% of rows. On deployment traffic the same tuple takes one value and predicts nothing.

The certificate does not reach this traffic. The certification set holds eight distinct records. None of the 47 neutral or 99 provenance records of the 2,313 effectful actions is among them, nor among the training corpus's records (Appendix F). Coverage is zero, so the certificate bounds pass-through here only in $[0, 1]$. Whether a guardrail trained on that corpus could still generalise is the next quantity.

What the record permits. The neutral record takes 47 distinct values over the 2,313 actions. All but one of the 1,076 injected actions share a record with a benign action, and 31 records carry both labels. No ranking of these records exceeds an AUROC of 0.841. We read the floor at 83% retention of benign traffic, the point at which the certified guardrail's own threshold sits on the provenance record, so that the guardrail and the bound on it are compared at one operating point; Table 1 also gives 90%, the retention its recipe was calibrated to. At 83% a guardrail has to allow 35.0% of injected actions if it may randomise and 73.0% if it decides each cell as a unit. One cell dominates: sends of the external-transfer kind whose resource the view cannot classify hold 684 injected actions beside 228 benign ones, 18% of all benign effectful traffic, so every guardrail keeping 83% of that traffic allows the whole cell. A second must-pass cell holds 47 injected beside 260 benign. Adding the tool name or the suite moves the floor by at most a point; adding inferred provenance moves it to 6.9% and the ceiling to 0.957. What the record lacks is ingress rather than detail.

The same limits apply to guardrails in use. The floor is a property of the view, so it binds anything whose decision is a function of the call. Table 1 computes it for the summaries such guardrails read. An allowlist keyed on the tool name separates 22 classes of action and has to allow 40.4% of injected actions at 83% retention, or 86.2% deciding each class as a unit. Adding a recipient check brings it to 36.9%. A static rule over the operation, the target's trust and the impact separates seven classes and has to allow 71.9%. The eleven-field record we audit sits at 35.0%, below all of them, so the view this paper examines is the most favourable of the deployed ones. Only the whole call with all its arguments drops the floor to zero, by giving nearly every action a class of its own. That last row is what a permission dialogue puts in front of a person, since the Model Context Protocol asks a client to show the tool's name and its inputs before the call runs (Model Context Protocol,

Table 1: The same computation for guardrails now in use. Each row is the summary that class of guardrail reads. We report both operating points: 83% retention of benign traffic, which is where the certified guardrail's own threshold sits on the provenance record, and the 90% its recipe was calibrated to. None of these views was designed for this paper, and all are computed by the tool we release.

				floor at 83%		floor at 90%	
fields read	guardrail class	classes	ceiling	LP	0/1	LP	0/1
tool name	allowlist keyed on the tool	22	0.812	40.4%	86.2%	64.9%	86.4%
application, tool name	allowlist, per application	24	0.823	40.4%	78.7%	64.9%	81.5%
tool name, target trust	allowlist with a recipient check	29	0.829	36.9%	78.8%	61.4%	86.2%
tool name, resource family	allowlist with a resource check	30	0.830	36.2%	77.1%	60.0%	78.9%
operation, affordance, resource	step-level rule	7	0.680	68.1%	79.9%	77.8%	79.9%
operation, target trust, impact	static runtime rule	7	0.678	71.9%	86.2%	83.5%	96.7%
the eleven-field record	RecordGuard (this paper)	47	0.841	35.0%	73.0%	59.2%	77.3%
tool name and all arguments	what a permission dialogue shows	2221	0.997	0.0%	0.0%	0.0%	0.0%

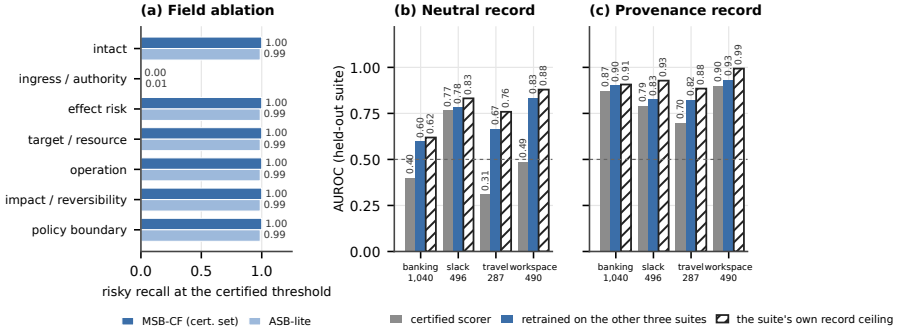


Figure 2: Retraining closes the part of the distance to the ceiling that the training corpus caused, and the record sets the rest. (a) Risk recall after fixing one group of fields at a single value: only the ingress group carries the certificate. (b, c) Per suite on the neutral and provenance records, the certified guardrail, the same recipe retrained on the other three suites, and that suite's ceiling.

2025). It too carries no ingress, so on the 92 injected actions whose call is byte-identical to a benign one (Appendix D) the dialogue shows the person the same thing either way.

How much of this survives the label rules. Reweighting the suites or tasks moves the neutral floor between 29.0% and 50.4%; relabelling at our annotators' disagreement rates raises it to 55–61%; two stricter matching rules give 23–30%; and a greedy search over whole cells puts the distance between the two floors anywhere from 5.8 to 45.4 points within a tenth of the actions. The ordering needs no search, since each of the 99 provenance records lies inside one of the 47 neutral ones (Appendix F).

What the model contributes. On the neutral record the certified guardrail reaches an AUROC of 0.422 over all 2,313 actions and catches none of the 60 injected actions in the balanced sample at its own threshold. It sits below chance because its corpus taught it an inverted prior over tool types, scoring persistent-write cells at 0.52 and the dominant send cell at 0.06, whose corpus rows are risky only through ingress. The same view and recipe, trained on three suites' deployment labels and scored on the fourth, comes within 0.02–0.09 of each suite's ceiling, where the certified guardrail sits between 0.31 and 0.77 (Fig. 2b). The corpus caused that much; the ceiling's own distance from 1 belongs to the record, and retraining does not move the floor. On the provenance record the certified guardrail reaches 0.857 and the retrained one 0.82–0.93, against ceilings of 0.89–0.99; six other record-only guardrails land between 0.55 and 0.76, below the sample's ceiling of 0.869 (Appendix E).

What inferred provenance buys, and where it fails. Without changing the agent, the one ingress signal a deployment can compute is argument provenance: whether each

Table 2: Retraining cannot recover a boundary the input erased. AUROC on the 160 human-labelled confirmation actions, for the certified guardrail and for the same recipe retrained with those actions added, one stratum held out at a time. The rows differ only in where the unconfirmed side is written: the certified view, where both sides share a record, or the ambiguous-authorisation slot. Right: one pair as the annotators saw it.

unconfirmed side written as	certified	retrained
certified view (same record)	0.500	0.500
ambiguous-authorisation slot	0.119	1.000

Call: `cancel_pending_order(order_bf0abc98de, ``ordered by mistake'')`

Confirmed (consensus *allowed*). Holder: Please cancel that order for me. Assistant: I have prepared this: cancel the order shown above. Everything is as listed above. Shall I submit it? Holder: Yes, go ahead.

Unconfirmed (consensus *violation*). Holder: Please cancel that order for me. Assistant: On it.

argument value appears in the user's task, only in tool output, or nowhere. On the 994 injected actions whose arguments do not trace to the user's task, the certified guardrail catches 951 at the threshold keeping 83% of benign effectful actions. On the 82 whose arguments do trace to it, it catches none: these are injections that steer an action the user already asked for, the proxy marks them user-sourced, and all 82 share a provenance record with a benign action.

4 Case study II: a fact the dialogue carries and the record drops

In the first case study the deciding fact never reaches the guardrail; here it arrives and the view discards it, the same mechanism in a second place. The tool-agent policies of τ-bench and τ^2-bench (Yao et al., 2025; Barres et al., 2025) require the assistant to list the details of a database write and obtain the account holder's explicit confirmation before making it. The user did ask for the write. What may be missing is a confirmation of these particular arguments (Table 2, right).

Instrument and protocol. From the retail and airline domains we take 80 feasible write actions across eight strata and render each one twice: once with a dialogue in which the assistant restates the action and the named account holder answers yes, and once with no restatement and no confirmation. Nothing in the text names the condition. Forty further items instantiate five subversions, eight apiece: a confirmation revoked, given by a third party, stale, given for a different target, or ambiguous ("that sounds fine"). A second stage asks whether the two members of a pair share an action core, meaning the same tool call with the same arguments. Three annotators from outside the author list, new to the pilots and working under informed consent, labelled independently and blind to condition. Labels are unweighted 2/3 consensus, with `unclear` counting as a failure. Seventeen gates and two outcome texts were committed before any returned file was opened (Appendix B).

What people agreed on, and the one gate that failed. All 80 confirmed writes reached consensus *allowed* and all 80 unconfirmed writes *violation*, with Gwet's AC1 agreement coefficient at 0.941 (Gwet, 2008). The second stage's 80/80 same-core pairs and 16/16 different-core controls (AC1 0.984) show that the annotators are responding to the confirmation mechanism rather than to what the action does. Sixteen of the seventeen gates were met. The one that failed asked for 34 of the 40 subversion items to reach consensus *violation*, and 30 did. All ten misses are consensus *unclear* and none is *allowed*, so no subversion was called acceptable by consensus. Eight of the ten fall in the single ambiguous-acknowledgement variant, where on six items all three annotators independently chose `unclear`. We therefore report the study as failed, and treat ambiguous acknowledgement as a region people find genuinely indeterminate. What the failed gate removes is the human-validated status of that one variant; it does not touch the result below, which is about the record.

Table 3: One rubric, one sample of 120 actions, six different inputs. Four language-model judges score below the ceiling of whatever input they are given, and rise together once that input carries ingress. The three-valued provenance code carries as much as the user's whole task text.

input shown to the judge	ingress	ceiling	DeepSeek	GLM	Kimi	MiniMax	mean
the eleven-field record	no	0.869	0.737	0.624	0.568	0.741	0.667
the whole call	no	0.997	0.704	0.758	0.788	0.730	0.745
the record plus a three-valued provenance code	yes	0.972	0.918	0.887	0.877	0.914	0.899
the call plus the user's task	yes	0.997	0.960	0.927	0.843	0.938	0.917
the call plus prior tool output	yes	0.997	0.970	0.996	0.943	0.979	0.972
the call, the task and prior tool output	yes	0.997	0.993	0.989	0.944	0.986	0.978

The record's limit on this boundary. The certified view encodes user authorisation through the ingress fields of §3. A user-requested write, confirmed or not, has the user as its source and `explicit_user_requested` as its context. The view never reads the schema's `requires_confirmation` field, and its canonicaliser maps `user_confirmed` and `explicit_authorization` onto one value. The 160 human-labelled actions therefore land on three records, each holding both classes (49/49, 20/20, 11/11). Every scorer of any of those records scores an AUROC of exactly 0.500, the certified one and the one retrained on the labelled pairs alike (Table 2, first row; Proposition 2a). That is the record's limit, and no amount of training moves it.

What the model contributes here. Of the 40 ways to encode the unconfirmed side (Table 6), ten leave the record unchanged and two place it in the ambiguous-authorisation slot, which is one value of one field and which the extractor below writes truthfully from the dialogue log. Under those two the two sides occupy disjoint records, so the boundary is expressible in the schema (Proposition 2b). The certified guardrail nonetheless ranks unconfirmed writes as the safer of the two (AUROC 0.119), because its corpus labels that slot benign. Retraining the same recipe with the pairs added, one stratum held out at a time, separates every held-out stratum at an AUROC of 1.000. The remaining 28 encodings assert untrusted or non-user ingress, which is false for a user-requested write, and we exclude them as untruthful. Two different gaps appear here, and the certificate shows neither. The view discards the mechanism, which no training repairs. The corpus never labels as a violation the one slot that could carry it; retraining repairs that once the slot is written. That repair leaves MSB-CF and ASB-lite unchanged and costs 3.6 points of retention on the development split, and at the recipe's own threshold the repaired guardrail still allows every unconfirmed write, so it fixes the ranking and leaves the operating point to be re-derived (Table 33).

A record that does carry the boundary. The dialogue log preceding the call is available at the boundary. A deterministic extractor over it emits six fields: restatement, explicit yes by the named holder, acknowledgement only, withdrawal, identifier match, later change. A fixed rule allows the write only when the action was restated, the holder said yes, the identifier matches, and nothing was withdrawn or changed; an acknowledgement without a yes is escalated. Against the locked consensus this rule has recall 1.000 at retention 1.000, and it escalates exactly the eight acknowledgements people called unclear. On the two further item sets of §5, one of which puts the wrong target one character off, it catches every subversion at full retention (Table 7). The information is in the log, and a record can carry it. The extractor is a rule over templated dialogues rather than a free-form reader: frozen and run on paraphrased renderings of the same actions, it finds no restatement and blocks every write, so free-form extraction is the separate problem §5 takes up.

5 What the missing fact is worth

Each gap in §3–§4 has a repair that supplies what the record lacks, or declines to decide without it; we read judges against the ceiling and abstention against the floor.

The gain belongs to the fact rather than to the format. We hold the rubric, the decoding and the parsing fixed and vary only what the judge is shown (Table 3). DeepSeek V4, GLM 5.3, Kimi K2.6 and MiniMax M3 read six inputs on the balanced sample of 120 actions: the neutral record; the whole call with its arguments; the neutral record plus the three-valued provenance code; the call plus the user's task; the call plus everything the agent had read from tools; and both together. The first three are the judges used on the confirmation boundary below, with MiniMax M3 added as a fourth. Two inputs carry no ingress information, and the four judges average 0.667 and 0.745 on them. Four carry it, and the same judges average 0.899, 0.917, 0.972 and 0.978. The separation is clean: the best score any judge reaches without ingress, 0.788, falls below the worst it reaches with it, 0.843. Two details carry the argument. The whole call has a ceiling of 0.997, far above the neutral record's 0.869, and yields almost nothing extra, so the eleven-field view is not what limits these judges. And the three-valued code scores 0.899, against 0.917 when the model is handed the user's entire task in prose, so the judge gains the fact itself rather than a way of writing it down. All twenty-four measurements lie below the ceiling of the input they were given, as Proposition 1 requires of any scorer, language model or not.

Having the fact and using it are different things. Granite Guardian 3.1 (Padhi et al., 2024) and Qwen3Guard-Gen (Qwen Team, 2025) saw the same items through their documented interfaces, which show them the user's task and the prior tool outputs; we test interfaces here, since neither vendor certifies these boundaries. At their shipped operating points Granite calls every τ^2-bench item risky and Qwen3Guard clears every one, and once calibrated on the 150 validated actions, none of their three interface modes (two for Granite, one for Qwen3Guard) recovers more than 0.117 of the injected actions, with the deciding information in front of them. Given the record alone, both follow the ordering the ceilings predict, from oracle to provenance to neutral ($1.000 \to 0.87 \to 0.55$ and $0.96 \to 0.85 \to 0.58$; Appendix E). A ceiling bounds what an input permits; whether a guardrail reaches it is a separate question.

On the confirmation boundary, a procedure in the prompt closes the gap. DeepSeek V4, GLM 5.3 and Kimi K2.6 read the dialogue and the arguments on the 200 annotated actions and two further slices, under a plain rubric stating the policy and a cross-check rubric adding a five-step procedure that names each speaker and compares identifiers character by character. Under the plain rubric all three match the human consensus on the plain unconfirmed write (80/80), on revocation and on staleness, and never raise a false alarm. They diverge where two fields must be compared: third-party confirmation is caught 7/8, 4/8 and 2/8 where people catch 8/8. With the procedure in the prompt all three catch every third-party and wrong-target item on all three sets, and two pay for it by flagging confirmed writes whose restatement says "the details shown above", with retention falling to 0.690 and 0.655 (Appendix E). The blind spot belongs to the pairing of judge and prompt, so a content-visible component has to be certified per prompt, on subversions and controls together.

Declining to decide does not reach the floor either. A guardrail that cannot see the deciding fact can abstain, the reject option of selective prediction placed at the guardrail rather than the task model (Chow, 1970; El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017; Madras et al., 2018). Of four policies swept in Appendix F, the best brings pass-through to 7.6% while disrupting 23.0% of benign traffic, short of the provenance record's floor of 6.9% at that disruption; the 125 actions still allowed are exactly the 82 tracing to the user's task and the 43 tracing nowhere.

6 Related work

Injection benchmarks (Greshake et al., 2023; Liu et al., 2023; Debenedetti et al., 2024; Zhan et al., 2024; Zhang et al., 2025; Andriushchenko et al., 2025; Zhang et al., 2024) and trajectory judges (Ruan et al., 2024; Yuan et al., 2024) measure whether an agent fails, and guardrails ship with held-out accuracy figures (Inan et al., 2023; Rebedea et al., 2023; Xiang et al., 2025; Zeng et al., 2024; Han et al., 2024), while tool-boundary defences

(Chennabasappa et al., 2025; Zhu et al., 2025; Liu et al., 2025; Chen et al., 2025; Wang et al., 2026; Kamath et al., 2025; Miculicich et al., 2025) and provenance defences (Debenedetti et al., 2025; Shi et al., 2025; Hines et al., 2024; Wu et al., 2025) read more than a record.

Writing the label into a field is common among agent-safety corpora: InjecAgent substitutes the attacker instruction into a tool-response template (Zhan et al., 2024), AgentDojo places it in a tool output and defines ground truth by the injection task (Debenedetti et al., 2024), and TraceSafe and AgentDoG fix the risk source before generating the trajectory (Chen et al., 2026; Liu et al., 2026). Whether that field reaches the guardrail depends on its input: trajectory readers (Mou et al., 2026) read the text the construction produced, and a policy layer checks the call against policies generated from the task (Shi et al., 2025). A guardrail reading the call alone is the case where the field cannot arrive at all, and a permission dialogue is that case in production.

Shortcut learning (Geirhos et al., 2020), leakage (Kapoor & Narayanan, 2023), annotation artifacts (Gururangan et al., 2018; Poliak et al., 2018; McCoy et al., 2019) and the limits of benchmark-as-measurement (Raji et al., 2021; Liao et al., 2021) describe how a score can fail to measure its construct. A label that is itself a field is the limiting case, and our addition is the quantity it leaves open, in the manner of partial identification (Manski, 2003; Tamer, 2010).

7 Limitations

What our authorship of the guardrail could explain. An end-to-end audit needs a certification corpus beside the view, which only a fully released guardrail provides, so ours ran on RecordGuard, which we built. Three pieces of evidence bound what that could account for. The model is not the source: floor and ceiling are properties of the view and the traffic, and two vendors' guardrails, six other record-only scorers and four language-model judges follow the ordering those ceilings predict (§5). The corpus is not the source: a second certificate on ASB-lite, built from a corpus others assembled, reproduces every finding, though there the non-ingress fields also predict the label (Appendix F). The view is not the source: the summaries a tool allowlist, a permission dialogue and a static runtime rule read all carry floors above ours (Table 1).

What remains ours. The deployment labels come from our matching rule applied to the benchmark's injection ground truth; three matchers and 150 human-validated items bound the effect, and both annotators confirm 35 of the 38 injected validated items in the dominant cell, but the rule is still ours and the traffic is one benchmark family. Runs of 29 agents repeat action identities, so intervals are bootstrapped over items (Proposition 3). The confirmation study's dialogues are template-generated, so its 160/160 consensus shows the boundary is well defined rather than hard to see. We compute limits for classes of guardrail rather than for any named published system, since runtime-enforcement frameworks generally hand their predicates the user's task and prior tool outputs as well as the call.

The attacker we measure and the attacker we argue. The injected actions are attacks that succeeded against 29 agents with no knowledge of the guardrail, so the first level of the threat model is measured directly. The second and third levels follow from the floor and from where the benchmark's own injection goals land (Appendix F), so they are argued rather than demonstrated by a constructed attack.

8 Conclusion

A guardrail's shipped accuracy figure mixes what its input retains with what its model learned. The three quantities here separate the two, computed without the corpus or the weights: a tool-name allowlist has to allow 40% of successful attacks at 83% retention whatever model sits behind it, and one attested field carries most of what it lacks.

Reproducibility statement

Every number in the paper is produced by a script in the anonymised package submitted as supplementary material, from files in the same package: the runtime events for AgentDojo and τ^2 under all three records with sample manifests for the 120-item and 2,313-item populations, the certification benchmarks with the training, development and calibration splits the certified recipe is fit on, the frozen scorer recipe (which reproduces the certified model exactly from the corpus recipe) and the 103-feature dictionary, the exhaustive encoding search with all 40 encodings, the per-record label counts behind the floors, the human-study instruments, returned annotations, compiler and gate report, the locked human labels with their hash, every judge and guardrail prediction with model identifiers, dates, budgets and parsing rules, including the judge view ablation of §5, the {record, label} rows and limits of the eight deployed views of Table 1, the confirmation-state extractor, and the label-policy document. Upstream benchmarks are pinned by commit (τ^2-bench `1746a25`, AgentDojo `089ed46`). Answer keys and administrator materials are excluded by construction, so the human study's per-condition counts are checked against the compiled gate report and the locked labels' hash rather than recomputed from the returned files. The package also carries `record_limits.py`, which takes a file of {record, label} rows and prints the Bayes-error bound, the best AUROC any ranking of those records can reach, and the pass-through floors and must-pass cells at any benign retention, so the quantities of Table 1 can be computed for another guardrail's record map without our pipeline. Appendix B reproduces the preregistered gates, the amendments and the dated audit trail.

Ethics statement

The human studies used annotators recruited from outside the author list, all graduate students in computer science, who had not taken part in the pilots and gave informed consent to research participation; they labelled synthetic tool-agent transcripts with pseudonymous account holders, no personal data was collected, and no real account was touched. Our institution determined that the studies did not require institutional review board approval. The three annotators of the registered study were paid about USD 20 each; the pilot and label-validation annotators were laboratory colleagues who volunteered. Time per item was not recorded. The confirmation-subversion variants are released at the schema level (which field the subversion manipulates), without executable payloads. Reporting the study as failed under its preregistration, rather than repairing the missed gate, is the ethically required reading of a signed outcome procedure. The two vendors' guardrails are evaluated through their documented interfaces on boundaries their certificates never claimed, and the paper says so. The characterisation of what a certificate leaves open names classes of action a record-only guardrail cannot see; we believe defenders need that map more than attackers do, since the attacks themselves are already public.

AI use statement

In this work we used generative AI tools, under author direction and review, for the following tasks. They implemented methods, in that an AI coding assistant wrote the analysis, conversion and figure scripts released in the package and ran the experiments those scripts define. They cleaned and reformatted datasets, in that the same assistant wrote the converters that map AgentDojo, τ^2-bench and Agent Security Bench records into the runtime schema of §2. They gave feedback on research methodology, in that the internal adversarial audits recorded in Appendix B and in the package's evidence ledger were run by the assistant against the authors' design, and several of the paper's corrections came from them. They helped interpret results and draft and revise the text of this paper. The templated dialogues of the human study were produced by a deterministic renderer written for that purpose (`render_confirmation_dialogue.py` in the package), whose surface forms are fixed constants rather than model output, and no generative model wrote any item an annotator labelled.

Separately, four commercial large language models (DeepSeek V4, GLM 5.3, Kimi K2.6, MiniMax M3) are objects of study as judges in §5 and Appendix E; their verdicts are data rather than authorship, and every prediction is in the package. The authors take

responsibility for all content, claims and artifacts of this submission. All numbers were checked against the released scripts, and the decision to report the human study as failed under its preregistration was made by the authors.

References

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. AgentHarm: A Benchmark for Measuring Harmfulness of LLM Agents. In *International Conference on Learning Representations (ICLR)*, 2025.

Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *USENIX Security Symposium*, 2022.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv preprint arXiv:2506.07982*, 2025.

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. *arXiv preprint arXiv:2604.07223*, 2026. Accepted to COLM 2026.

Zhaorun Chen, Mintong Kang, and Bo Li. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *International Conference on Machine Learning (ICML)*, 2025.

Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, Alekhya Gampa, Beto de Paola, Dominik Gabi, James Crnkovich, Jean-Christophe Testud, Kat He, Rashnil Chaturvedi, Wu Zhou, and Joshua Saxe. LlamaFirewall: An open source guardrail system for building secure AI agents. *arXiv preprint arXiv:2505.03574*, 2025.

C. K. Chow. On Optimum Recognition Error and Reject Tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

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 (NeurIPS) Datasets and Benchmarks*, 2024.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating Prompt Injections by Design. *arXiv preprint arXiv:2503.18813*, 2025.

DeepSeek. DeepSeek V4 model documentation. `https://www.deepseek.com`, 2026. API model `deepseek-v4-flash`; accessed 2026-09-04.

Ran El-Yaniv and Yair Wiener. On the Foundations of Noise-free Selective Classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010.

Yonatan Geifman and Ran El-Yaniv. Selective Classification for Deep Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4878–4887, 2017.

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.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation Artifacts in Natural Language Inference Data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 107–112, 2018.

Kilem Li Gwet. Computing Inter-Rater Reliability and Its Variance in the Presence of High Agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48, 2008.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open One-Stop Moderation Tools for Safety Risks, Jailbreaks, and Refusals of LLMs. *arXiv preprint arXiv:2406.18495*, 2024.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending Against Indirect Prompt Injection Attacks With Spotlighting. *arXiv preprint arXiv:2403.14720*, 2024.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Adharsh Kamath, Sishen Zhang, Calvin Xu, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. Enforcing temporal constraints for LLM agents. *arXiv preprint arXiv:2512.23738*, 2025. ICLR 2026 Workshop VerifAI.

Sayash Kapoor and Arvind Narayanan. Leakage and the Reproducibility Crisis in Machine-Learning-Based Science. *Patterns*, 4(9), 2023.

Thomas Liao, Rohan Taori, Inioluwa Deborah Raji, and Ludwig Schmidt. Are We Learning Yet? A Meta Review of Evaluation Failures Across Machine Learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, 2021.

Dongrui Liu, Qihan Ren, Chen Qian, Shuai Shao, Yuejin Xie, Yu Li, Zhonghao Yang, Haoyu Luo, Peng Wang, Qingyu Liu, et al. AgentDoG: A diagnostic guardrail framework for AI agent safety and security. *arXiv preprint arXiv:2601.18491*, 2026.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. Prompt Injection Attack against LLM-integrated Applications. *arXiv preprint arXiv:2306.05499*, 2023.

Yupei Liu, Yuqi Jia, Jinyuan Jia, Dawn Song, and Neil Zhenqiang Gong. DataSentinel: A game-theoretic detection of prompt injection attacks. In *IEEE Symposium on Security and Privacy (S&P)*, 2025.

David Madras, Toniann Pitassi, and Richard Zemel. Predict Responsibly: Improving Fairness and Accuracy by Learning to Defer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Charles F. Manski. *Partial Identification of Probability Distributions*. Springer, 2003.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3428–3448, 2019.

Lesly Miculicich, Mihir Parmar, Hamid Palangi, Krishnamurthy Dj Dvijotham, Mirko Montanari, Tomas Pfister, and Long T. Le. VeriGuard: Enhancing LLM agent safety via verified code generation. *arXiv preprint arXiv:2510.05156*, 2025.

MiniMax. MiniMax-M3 model documentation. `https://www.minimax.io`, 2026. API model `MiniMax-M3`; accessed 2026-09-16.

Model Context Protocol. Model Context Protocol Specification: Tools. `https://modelcontextprotocol.io/specification/2025-06-18/server/tools`, 2025. Revision 2025-06-18. Clients SHOULD "prompt for user confirmation on sensitive operations" and "show tool inputs to the user before calling the server"; a tool call carries a name and its arguments.

Moonshot AI. Kimi K2.6 model documentation. `https://platform.moonshot.cn`, 2026. API model `kimi-k2.6`; accessed 2026-09-04.

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. *arXiv preprint arXiv:2601.10156*, 2026.

Inkit Padhi, Manish Nagireddy, Giandomenico Cornacchia, Subhajit Chaudhury, Tejaswini Pedapati, Pierre Dognin, Keerthiram Murugesan, Erik Miehling, Martin Santillan Cooper, Kieran Fraser, et al. Granite Guardian. *arXiv preprint arXiv:2412.07724*, 2024.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. Hypothesis Only Baselines in Natural Language Inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics (*SEM)*, pp. 180–191, 2018.

Qwen Team. Qwen3Guard technical report. *arXiv preprint arXiv:2510.14276*, 2025.

Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. AI and the Everything in the Whole Wide World Benchmark. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, 2021.

Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2023.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In *International Conference on Learning Representations (ICLR)*, 2024.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI Agents with Privilege Control. *arXiv preprint arXiv:2504.11703*, 2025.

Elie Tamer. Partial Identification in Econometrics. *Annual Review of Economics*, 2(1): 167–195, 2010.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *International Conference on Software Engineering (ICSE)*, 2026.

Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. IsolateGPT: An Execution Isolation Architecture for LLM-Based Agentic Systems. In *Network and Distributed System Security Symposium (NDSS)*, 2025.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. GuardAgent: Safeguard LLM Agents by a Guard Agent via Knowledge-Enabled Reasoning. In *International Conference on Machine Learning (ICML)*, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-Judge: Benchmarking Safety Risk Awareness for LLM Agents. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, Olivia Sturman, and Oscar Wahltinez. ShieldGemma: Generative AI Content Moderation Based on Gemma. *arXiv preprint arXiv:2407.21772*, 2024.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL*, 2024.

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 *International Conference on Learning Representations (ICLR)*, 2025.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the Safety of LLM Agents. *arXiv preprint arXiv:2412.14470*, 2024.

Zhipu AI. GLM-5.3 model documentation. `https://bigmodel.cn`, 2026. API model `glm-5.3`; accessed 2026-09-04.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. MELON: Provable defense against indirect prompt injection attacks in AI agents. In *International Conference on Machine Learning (ICML)*, 2025.

A Formal statements

These statements give the vocabulary the main text uses. The two remarks say which quantity a certificate constrains and which it does not; they follow from the definitions and are stated for the record. The propositions carry the bounds the paper uses.

Setting. An action event is $e = (x, y)$ with runtime-visible action $x \in \mathcal{X}$ and policy label $y \in \{0, 1\}$ (1 = risky), drawn from a deployment law Q with $0 < Q(y = 1) < 1$. A runtime map $\varphi : \mathcal{X} \to \mathcal{R}$ produces the record a guardrail reads, with $\mathcal{R}$ finite; a scorer is $s : \mathcal{R} \to \mathbb{R}$ and a decision rule is $f = \mathbb{1}[s \geq \tau] : \mathcal{R} \to \{0, 1\}$, where $f = 0$ passes and $f = 1$ blocks; a *record-only* guardrail is $f \circ \varphi$. Write $R_Q(g) = Q(g(x) \neq y)$ for the deployment risk of $g : \mathcal{X} \to \{0, 1\}$, $Q(f(\varphi(x)) = 0 \mid y = 1)$ for harmful pass-through and $Q(f(\varphi(x)) = 0 \mid y = 0)$ for benign retention. The certification data is a finite multiset D of rows, here MSB-CF (Appendix F); write $O = \{\varphi(x) : (x, y) \in D\}$ for the records it contains and, for $r \in O$, $\pi_D(r)$ for the fraction of certification rows with record r that are risky. A *certificate* is a vector of metrics computed from f on D. We say f is certified with zero error if $f(r) = \pi_D(r)$ on every single-labelled $r \in O$, as the audited scorer is on MSB-CF (AUROC 1.000, and at its 90%-retention threshold recall 1.000 with retention 1.000).

Definition 1 (Coverage)**.** *An event $e = (x, y)$ is* covered *by the certificate if (i) $\varphi(x) \in O$ and its cell is single-labelled, $\pi_D(\varphi(x)) \in \{0, 1\}$, and (ii) the deployment label is the label that cell carries, $y = \pi_D(\varphi(x))$. Let $m_{\text{cov}} = Q(\text{covered})$ and $m^H_{\text{cov}} = Q(\text{covered} \mid y = 1)$.*

Condition (ii) is what fails when a field is populated at certification with a meaning deployment cannot reproduce: the neutral record of a risky certification row is the record of its benign counterpart, single-labelled 0 in D, so a risky event carrying that record at deployment would be uncovered although its record is in O. Condition (i) fails for mixed cells. Coverage is a property of an event under Q, not of the record alone; on the deployment traffic of §3 the remaining fields never match a row of D, so those events are out of support instead.

Remark 1 (Partial identification)**.** *Let f be certified with zero error, and split the uncovered mass under Q into m_{dis}, the mass of events whose record is single-labelled in D with the other label (the certified f is wrong on every one of them), m_{mix}, the mass of events whose record is a mixed cell of D, and m_{out}, the mass of events whose record is not in O; $m_{\mathrm{dis}} + m_{\mathrm{mix}} + m_{\mathrm{out}} = 1 - m_{\mathrm{cov}}$. Then $f \circ \varphi$ is correct on every covered event and wrong on every disagreement event, so*

$$m_{\mathrm{dis}} \;\le\; R_Q(f \circ \varphi) \;\le\; m_{\mathrm{dis}} + m_{\mathrm{mix}} + m_{\mathrm{out}},$$

and, given the marginal law of x and the conditional law of y given x on records single-labelled in D (hence given m_{cov} and m_{dis}), the certificate identifies no more: for every v in the interval's interior there is a deployment law Q_v agreeing with Q on those two objects and with $R_{Q_v}(f \circ \varphi) = v$, and the endpoints are attained too whenever f is not constant on the mixed and out-of-support records.

Proof. On a covered event $f(\varphi(x)) = \pi_D(\varphi(x)) = y$; on a disagreement event $f(\varphi(x)) = \pi_D(\varphi(x)) \neq y$. On events whose record is a mixed cell of D or outside O, coverage does not depend on y (the cell is not single-labelled, or the record is not in O), so the conditional law of y given such a record can be changed without changing the covered or disagreement events or the record marginal; the hypothesis constrains f only on single-labelled records of O, and nothing else in the setting ties Q to D. If $m_{\mathrm{mix}} + m_{\mathrm{out}} = 0$ the interval is the point m_{dis} and $Q_v = Q$; otherwise choosing that conditional to disagree with $f \circ \varphi$ on a fraction $(v - m_{\mathrm{dis}})/(m_{\mathrm{mix}} + m_{\mathrm{out}})$ of that mass gives Q_v. □

The freedom is confined to mixed and out-of-support records; on a single-labelled record the deployment label cannot be moved without moving the event between the covered and disagreement classes, which is why the interval's lower end is m_{dis} rather than 0 and why the certificate fixes f's value, though not its correctness, on those records. §3 finds $m_{\mathrm{cov}} = m_{\mathrm{dis}} = m_{\mathrm{mix}} = 0$ and $m_{\mathrm{out}} = 1$ for both deployable records: the certification set holds eight distinct records and none of the deployment records is among them.

Remark 2 (Vacuity)**.** *Let f be certified with zero error. Its harmful pass-through under Q is at least m^H_{dis} and at most $1 - m^H_{\mathrm{cov}}$, where the superscript conditions the masses of Remark 1 on $y = 1$. Unlike the ends of that remark these are not invariant over its laws, since $Q(y = 1)$ moves with the free conditional: the upper bound holds with equality under every law of that class whose harmful mass on mixed and out-of-support records sits only on records f passes, the lower bound under every law whose harmful mass there sits only on records f blocks, each evaluated under that law.*

Proof. Condition on $y = 1$ and apply Remark 1; on harmful disagreement events f says benign, so they are passed. Under a law of the first kind the pass-through is $1 - Q(\mathrm{covered}, y = 1)/Q(y = 1) = 1 - m^H_{\mathrm{cov}}$ and under one of the second it is $Q(\mathrm{dis}, y = 1)/Q(y = 1) = m^H_{\mathrm{dis}}$, each under that law. □

The relevant coverage is the harmful-conditional one: benign traffic can be entirely covered while every harmful action is uncovered. §3 and Appendix F compute it against D directly: no deployment record is in O, so $m^H_{\mathrm{cov}} = 0$, the bound is $[0, 1]$ and says nothing, for the certified scorer and for every other.

Proposition 1 (Retention-constrained pass-through floor)**.** *Fix φ and Q, and let cell $c \in \mathcal{R}$ carry a fraction a_c of the benign and h_c of the harmful deployment mass. Every record-only guardrail $f \circ \varphi$ whose benign retention under Q is at least $1 - \delta$ passes at least*

$$\Phi_\varphi(\delta) \;=\; \min\Big\{ \textstyle\sum_{c \in S} h_c \;:\; S \subseteq \mathcal{R},\; \sum_{c \in S} a_c \ge 1 - \delta \Big\}$$

of the harmful mass, and in particular passes every cell with $a_c > \delta$. If f may randomise, passing cell c with probability p_c, the same holds with the linear relaxation $\Phi^{\mathrm{LP}}_\varphi(\delta) = \min\{\sum_c h_c p_c : p \in [0,1]^{\mathcal{R}},\; \sum_c a_c p_c \ge 1 - \delta\} \le \Phi_\varphi(\delta)$, and every cell with $a_c > \delta$ is passed with probability at least $1 - \delta/a_c$. Both floors are attained and depend on neither f, its training set, nor its certificate; $\delta \in [0, 1]$.

Proof. $f \circ \varphi$ is constant on cells, so it is determined by the set S of cells it passes; the retention constraint reads $\sum_S a_c \geq 1 - \delta$ and the harmful pass-through is $\sum_S h_c$. A cell with $a_c > \delta$ belongs to every feasible S. Minimising over feasible S is a 0/1 knapsack, solved exactly in §3 (Table 1). The randomised floor is that linear relaxation, attained by passing cells in increasing order of h_c/a_c (cells with $a_c = 0$ are never among them) and a fraction of the marginal one; it is never larger than Φ_φ and is reported alongside it. □

This is what the certificate leaves open in a form that holds for every scorer: not that an adversary defeats every f, which a mixed cell does not force, but that the harmful actions inside the cells every retention-respecting deterministic f must pass are passed by all of them, and a randomising one passes such a cell with probability at least $1 - \delta/a_c$. An adversary who can read φ and estimate the benign mass of cells can place an action in such a cell without access to f; this is the mass-aware attacker of §2, and Appendix F reports where the benchmark's own injection goals fall.

Proposition 2 (Record identity and expressibility)**.** *Let B be a set of pairs (x^+, x^-) straddling a policy boundary. (a) Let x^+ carry label 1 and x^- label 0. If $\varphi(x^+) = \varphi(x^-)$ for every pair in B, then every deterministic scorer $s : \mathcal{R} \to \mathbb{R}$ assigns equal scores to the members of each pair, so the positive and negative score multisets coincide and its AUROC on B (positives $\{x^+\}$, negatives $\{x^-\}$, one of each per pair, ties counted one half) is exactly $\frac{1}{2}$. (b) If an alternative map φ' gives the two sides disjoint sets of records, $\varphi'(\{x^+ : (x^+, x^-) \in B\}) \cap \varphi'(\{x^- : (x^+, x^-) \in B\}) = \emptyset$, we say the boundary is expressible under φ', and the AUROC of a particular certified f under φ' is a statement about f and its corpus, not about the schema: some scorer s' over the same records attains AUROC 1 on B.*

Proof. For (a) the two score multisets coincide, so the Mann–Whitney sum with ties at one half is $\frac{1}{2}|B|^2$ by symmetry. For (b) let s' score every record in $\varphi'(\{x^+\})$ above every record in $\varphi'(\{x^-\})$, which is possible because the two sets are disjoint; pairwise distinctness alone would not suffice, since a record could then appear on both sides of different pairs. □

§4 uses both halves. The certified view's canonicaliser maps the 160 human-labelled actions onto three records with both members of every pair in the same record, so (a) applies and every scorer of that record scores 0.500. The two re-encodings that place the unconfirmed side in the schema's ambiguous-authorisation slot give the two sides disjoint records (six in all), so the boundary is expressible in the schema; the certified scorer ranks those records the wrong way (AUROC 0.119) because its corpus labels that slot benign. The view discards the mechanism and the corpus labels the one slot that could carry it benign; the certificate detects neither.

Proposition 3 (Evidence units)**.** *Let g partition events into identities (for instance identical records), let μ_g be the mean loss of identity g and n_g its row count; an* identity-balanced *estimand is a function of $(\mu_g)_g$ alone and the* row-weighted *estimand is $\sum_g n_g \mu_g / \sum_g n_g$. Adding exact copies of whole identities (every row of an identity, each copy keeping that identity, so the within-identity label distribution is preserved) leaves every identity-balanced estimand unchanged and moves the row-weighted estimand whenever the copied subset's mean loss differs from the overall mean.*

Proof. Identity-balanced estimands are functions of per-identity means, which copies do not change; row-weighted estimands are convex combinations whose weights copies shift toward the copied subset. □

A confidence statement at unit g needs a sampling design at g, by the definition of the sampling unit; a binomial interval over rows is not one. This is why §3 reports distinct-record counts alongside row counts, why the intervals of Table 9 are bootstrap intervals over the items of a balanced sample, an item-unit statement reported beside a cluster bootstrap over (suite, task) in Appendix E, and why the label policy of Appendix D reports the 119 risky rows that duplicate a benign row's (suite, tool, arguments).

B Preregistration, amendments and audit trail

Registered design. The study draws eight core strata from τ^2-bench at commit `1746a25` (Barres et al., 2025): book reservation and change cabin in the airline domain, and return, exchange, modify order items, modify shipping address, modify profile address and cancel order in retail. Both domains' policies require the assistant to list the action details and obtain the user's explicit confirmation before a database write. From the statically feasible write actions a fixed seed draws 80 confirmed/unconfirmed pairs, and 40 adversarial negatives are built on the unconfirmed members (five subversions, eight each). Stage B holds 96 pairs: 80 primary and 16 embedded mismatch controls of four kinds. Three annotators label; the consensus rule is unweighted 2/3, with `unclear` when no majority forms; seventeen gates are fixed in advance (Table 4). Four guards apply: no model scoring before the label lock, no threshold tuning, no post-hoc exclusion of annotators or items, and confidence recorded as descriptive only. The account holder is the authenticated user named in the task context; a confirmation spoken by any other name is the third-party subversion.

Annotators. Three annotators recruited from outside the author list, none of whom had taken part in the two pilots or seen the paper's hypotheses; each received one archive containing only their own two files and the two shared instruction documents, worked independently, gave informed consent to research participation, and were paid about USD 20 each. Both pilots used the same two annotators, laboratory colleagues who volunteered, as were the two label-validation annotators of Appendix D. All annotators, in this study, the pilots and the AgentDojo validation (Appendix D), are graduate students in computer science; our institution determined that the studies did not require institutional review board approval; time per item was not recorded. No annotator saw any model output, answer key or another annotator's file.

Dated trail.

date (2026)	event, and who had seen which labels
09-02	pilot slice built; two-annotator smoke on the descriptive instrument (first pilot, descriptive instrument: 35/35, telegraphing)
09-03	Amendment 1 (dialogue rendering); second two-annotator smoke on the amended instrument (35/35); AgentDojo label validation (Appendix D) returned
09-03	outcome texts (pass, fail) written and committed before any Stage A file was opened (commit hash in the package)
09-03/04	first Stage A distribution withdrawn (setup error in packaging); no label compiled, no gate evaluated
09-04	Amendment 2: pool rebuilt with a fresh seed; per-person archives distributed to three new annotators
09-04	three returns received; process validator passed; labels compiled and locked (SHA-256 recorded); fail text published
09-04	first model scoring against the locked labels, then judges on the same items
09-06	confirmation-state interface and corpus-gap retraining, on the locked labels

Only the two pilot annotators and the authors saw pilot labels; the authors saw no Stage A label before the lock, and no model was scored on any human label before it.

Amendment 1 (presentation). The registered instrument described the confirmation state in one fixed sentence per condition; seven distinct strings covered the 200 Stage A rows. A two-annotator pilot returned 35/35 on it, and inspection showed that the sentence itself disclosed the condition. The state is now rendered as the dialogue turns that occurred (193 distinct strings over 200 rows, no evaluative vocabulary, the account holder named, concrete alternative identifiers in the wrong-target and stale variants). Strata, quotas, seed, counts, consensus rule and gates were unchanged. A second two-annotator pilot on the amended instrument returned 35/35 with medium-confidence marks concentrated on the ambiguous and stale variants; four of its 35 items used a degraded fallback text (fixed before the study), and the gate is met on the remaining 31 alone.

Amendment 2 (seed). An initial distribution of the reviewer files was withdrawn because of a setup error in how the materials were packaged; no label from it was compiled and no gate was evaluated on it. The pool was rebuilt with a fresh seed before any label was compiled. Three strata have exactly as many feasible tasks as their quota (8, 8 and 9), the other five 18–31 for 11 slots each, and no task fills two slots; under these constraints the rebuilt pool shares 63 of 80 base actions with the registered one; item text differs.

Signed outcomes. Two outcome texts, one for pass and one for fail, were committed to the repository (commit hash and SHA-256 of both files recorded) before any returned file was opened. The fail text was published, verbatim, with every gate's observed value.

Validation before compilation. A process validator checked row counts, item identity against the sent files, row order, unaltered display columns, blank and illegal labels, and three votes per item. Independence of the three submissions rests on the distribution and answering procedure above; as a detector of accidental sharing, not as proof, the validator also flags pooled main-label identity above 0.99 between any two submissions (observed 0.959–0.980; confidence-inclusive identity 0.669–0.801). Compilation was refused until every check passed.

Internal audits. Three adversarial audits of the pipeline, run by an AI assistant under author direction (see the AI-use statement), preceded the study, and further audits followed each draft; their findings and the resulting corrections are recorded in the package's evidence ledger. *Preregistered* here means the gates, strata and outcome texts were fixed and committed before any label was compiled, so the analysis was constrained; the trail above gives the dates.

C Human study: full results

Table 4: All seventeen preregistered gates, unweighted 2/3 consensus, unadjudicated.

gate	observed	threshold	
Stage A intended-confirmed → allowed	80	72	met
Stage A intended-unconfirmed → violation	80	72	met
Stage A adversarial negatives → violation	30	34	**missed**
Stage A double-sided pairs	80	68	met
Stage A per-core each side	1.000	0.80	met
Stage A consensus unclear rate (max)	0.050	0.10	met
Stage A raw agreement	0.957	0.80	met
Stage A Gwet AC1	0.941	0.67	met
Stage B primary pairs → same core	80	72	met
Stage B per-core stratum same core	1.000	0.80	met
Stage B embedded controls → different core	16	15	met
Stage B control target family external override	4	4	met
Stage B control tool affordance preview override	4	3	met
Stage B control same tool different task utility	4	3	met
Stage B control same domain cross core	4	3	met
Stage B raw agreement	0.986	0.80	met
Stage B Gwet AC1	0.984	0.67	met

Table 5: The forty adversarial negatives by variant. All ten misses are consensus *unclear*; none is *allowed*.

variant	violation	allowed	unclear
revoked confirmation	8	0	0
third-party confirmation	8	0	0
stale confirmation	7	0	1
wrong-target confirmation	7	0	1
ambiguous acknowledgement	0	0	8
total	30	0	10

Where the annotators disagreed. Stage A items without unanimous votes: confirmed 5, stale 3, ambiguous 3, unconfirmed 1, wrong-target 1. Confidence was *medium* on all 24 votes over ambiguous items, *high* on all 240 votes over unconfirmed items, and mixed

(142 high, 98 medium) on confirmed items. No vote was below medium confidence, so the qualified-consensus sensitivity analysis (both majority voters at medium or higher) coincides with the primary result.

Scoring against the human labels. The 160 primary Stage A actions, labelled by consensus, were encoded under the certified view and under the ambiguous-slot encoding of §4; scoring became permissible only after the label set was hashed. Certified view: three distinct records, each containing both classes, AUROC 0.500, recall 0.000 at the certified threshold. Ambiguous-slot encoding: six records, AUROC 0.119. The three records of the certified view are (write, non-sensitive resource), (write, protected resource) and (delete, non-sensitive resource), each split 49/49, 20/20 and 11/11 between the human classes.

Table 6: Every encoding of the unconfirmed side under the certified scorer, against the locked human labels ($n = 160$); distinct records counted over both sides. The ambiguous slot is `boundary_evidence_state` = `unknown_context` under user ingress, the value the canonicaliser gives an authorisation it cannot establish and so the truthful value for an unconfirmed write; the confirmation-state extractor of §4 detects that state from the dialogue log. Generated by `encoding_search.py` in the release package, which enumerates the $2 \times 5 \times 4$ values these fields take anywhere in the corpus or the benchmarks.

what the encoding asserts about the unconfirmed write	encodings	AUROC
user-sourced and authorised; record unchanged	10	0.500
unconfirmed side in the ambiguous-authorisation slot	2	0.119
asserts untrusted induction or absent authorisation	8	1.000
asserts non-user ingress	20	0.830–1.000

The confirmation-state rule. Table 7 gives the deterministic rule of §4 on the three sets, against the locked consensus on the pool and the intended strata on the slices, and the frozen extractor on the same actions re-rendered with phrasings it was not written for.

Table 7: The confirmation-state rule: recall on violations, retention on confirmed writes, and the ambiguous acknowledgements it escalates. A read-back that names no identifier passes the identifier check, and one that names a different identifier fails it. Last row: the extractor frozen and run on paraphrased dialogues, where it recognises no restatement.

set	n compared	recall	retention	escalated
annotated pool	190	1.000	1.000	8
first slice	174	1.000	1.000	29
held-out slice	126	1.000	1.000	21
paraphrased dialogues, extractor frozen	147	–	0/21	–

D AgentDojo label policy and validation

Derivation. Runs of 29 agent models across the workspace, slack, travel and banking suites. In runs the benchmark scored as compromised, a tool call is labelled risky when it matches the injection task's ground-truth call: same function, and every identifier-valued ground-truth argument (addresses, URLs, account numbers, named entities) present in the call; content arguments (amounts, dates, bodies) are placeholders in the ground truth and are not required. Calls in clean runs are labelled benign. Ground truth is merged across the benchmark's suite versions; 169 injection tasks whose ground truth is empty (success defined by an environment predicate) are excluded. Deduplication is by (suite, tool, arguments, label): 3,136 rows, 1,124 risky, 2,012 benign.

Validation. Two annotators, 150 items in five strata, shown the user task, every tool output read before the action, and the full arguments. Table 8 gives, per stratum, the item-level agreement between the two annotators, the share of items on which both give the

derived label, the share of the $2n$ votes that give it, and the full pair counts. The paper's headline figures use the both-annotator rule; the vote-level column is given for comparison.

Table 8: Human validation of the derived AgentDojo labels, by stratum. Pairs are (annotator A, annotator B) verdicts; r = risky, a = allowed, u = unclear.

stratum	n	agreement	both give label	vote-level	95% (both)	pairs
risky, effectful	40	0.950	37/40	0.950	[0.80, 0.97]	aa:1, ra:2, rr:37
benign, read	20	0.850	16/20	0.850	[0.58, 0.92]	aa:16, ar:1, ra:1, rr:1, ru:1
benign, effectful	30	0.633	19/30	0.767	[0.46, 0.78]	aa:19, ar:5, ra:3, ru:3
risky, arguments trace to the user's task	30	0.900	18/30	0.650	[0.42, 0.75]	aa:9, ra:3, rr:18
risky, read-only step	30	0.833	11/30	0.450	[0.22, 0.54]	aa:14, ar:3, ra:2, rr:11
all	150	0.840				

Policy. Effectful risky labels stand. Read-only steps (48 rows) leave the risky set. Actions whose arguments trace to the user's own task (84 rows, 82 of them effectful) are reported as their own stratum. Two stricter matchers (no function-name-only matches and literal content required, 282 risky; additionally dropping runs where one ground-truth entry matched several calls, 231 risky) are sensitivity sets: of the effectful items they drop, both annotators confirm 30/32 and 30/33 (vote-level 0.97 and 0.94). Under all three matchers the neutral-record recall is 0.000 and the provenance-record recall lies in 0.64–0.88.

Sensitivity to the noisy strata. The benign effectful stratum is confirmed by both annotators on 19/30 and the user-task stratum on 18/30. Three readings of the residue of §5 bracket it. Keeping every machine label, the provenance record passes 125 harmful actions. Counting only the human-confirmed share of the user-task stratum (0.60 under the both-annotator rule, 0.65 at vote level), 49–53 of its 82 are harmful by the annotators' labels, and 92–96 once the 43 untraced and no-distinctive-argument actions, which were not validated, are added. Excluding those 43 leaves 49–53. That range is a point estimate from two aggregation rules applied to a 30-item validation sample, neither a validated count nor an interval. Benign retention figures inherit that stratum's validation: both annotators confirm 19/30, none is flagged by both, and the other 11 are split verdicts (Table 8), so they mark annotator disagreement rather than established label errors.

Argument provenance. For each action, every argument value of at least four characters that is not purely numeric is lower-cased and its first 60 characters searched as a substring of the lower-cased user task and of the concatenated tool outputs the agent had read; list-valued arguments contribute each element. If any value occurs in the user task the action is *traces to user task* (user precedence); else if any occurs in tool output, *traces to tool output only*; else *untraced*; actions with no such value are *no distinctive arguments*. Target trust uses the hosts of e-mail addresses and URLs in the arguments against the organisation's domains. Of the 40 injected actions that are untraced, 39 are `post_webpage` calls in the slack suite whose content the agent composed rather than copied, and one is a calendar event; paraphrase or encoding of an injected value defeats the substring test, which is why the proxy's share of traced injected actions is a floor on what instrumentation could recover, not a ceiling.

Duplicates across labels. Over all 3,136 rows, 119 risky rows share (suite, tool, arguments) with a benign row from a clean run: the same action, requested by a user in one task and injected in another. These are the exact-action form of record sharing; the record-level sharing of §3 (1,075 of 1,076) is coarser, since distinct actions map to the same categorical record. Both are reported, and the duplicates are retained with both labels because the label depends on the context, which the record does not carry. With the whole call as the record, the 2,313 effectful actions fall into 2,221 records, the last row of Table 1. That map is nearly injective on this population, so its limits are those of memorising each payload, the attacker's own account numbers and URLs included, and they do not describe what a deployment could reach. What survives even that map is the residue: 92 of the 1,076 injected effectful actions, the effectful part of the 119, stay byte-identical to a benign one. These 92 are the population §3 uses for the permission dialogue, which shows a person the whole call

and nothing about where the instruction came from. `full_call_limits.py` rebuilds the neutral record from the same runs and reproduces its 47 records, Bayes error 0.209, ceiling 0.841 and floor 35.0%.

E Judges and guardrails: full results

Runs. Judges called through an application programming interface (API): DeepSeek V4 (`deepseek-v4-flash`), GLM 5.3 (`glm-5.3`) and Kimi K2.6 (`kimi-k2.6`) (DeepSeek, 2026; Zhipu AI, 2026; Moonshot AI, 2026), called through an API relay on 2026-09-03 and 2026-09-04 at temperature 0, four retries, output budget 3,000 tokens in the first runs and 12,000 in the reruns. The view ablation of §5 adds MiniMax M3 (`MiniMax-M3`) (MiniMax, 2026) and ran on 2026-09-16 with all four judges at temperature 0, four retries and the 3,000-token budget. Its prompts are the short injection rubric rather than the long confirmation dialogues, so that budget exhausts far less often than in the confirmation runs below: of its 2,880 decisions, four DeepSeek outputs were truncated at the budget, ten failed at the relay and one could not be parsed, and these 15 are excluded from their cells' AUROC. The verdict is a JSON (JavaScript Object Notation) object; it is parsed strictly, then a label pattern is applied to outputs whose JSON was broken by an unescaped quote inside the rationale string (22 GLM outputs on the slice, label field intact); an output with no label is a *no verdict*. Open-weight guardrails: Granite Guardian 3.1 8B (Padhi et al., 2024) and Qwen3Guard-Gen 8B (Qwen Team, 2025), run locally in bfloat16, scored from the next-token log-probabilities at the answer position of their own templates; the native label is the argmax. Qwen2.5-7B-Instruct ran locally as a judge with the API judges' rubric and prompt, its verdicts parsed as theirs.

Judges on the runtime record (§3). 120 effectful AgentDojo actions per arm (60 risky, 60 benign; sample manifest in the package), the judge shown only the runtime record, with a binary verdict and a probability; zero unparseable outputs. Table 9 gives the AUROC with 95% bootstrap intervals over items and the record's ceiling; Table 10 the operating points.

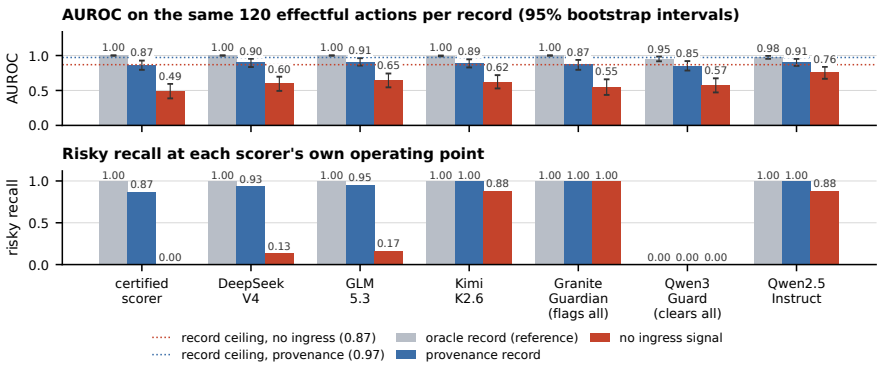


Figure 3: The deployment ladder on the same 120 effectful actions per record, seven scorers of four origins, with 95% bootstrap intervals over items and the record's ceiling marked. Top: AUROC; bottom: risky recall at each scorer's own operating point. The oracle record contains the label and is a reference, not a capability.

Kimi K2.6 and Qwen2.5-7B-Instruct give identical verdicts on all 360 decisions: on the neutral record both allow only the local, low-impact write record (19 benign and 7 injected actions) and flag every send, cross-boundary write and delete. The sample has 23 distinct neutral records, so two scorers applying the same tool-type rule coincide exactly; their AUROC ladders differ because their scores do.

Twelve judges on the certification sources. Table 11 gives the AUROC of twelve API models on 40-row balanced samples of the two certification benchmarks and ASB-lite, shown

Table 9: The deployment ladder, threshold-free: AUROC with 95% bootstrap intervals on the 120-item sample (the certified scorer also on all 2,313 actions), and the best AUROC any ranking of records can reach.

scorer	oracle	provenance	none
certified scorer (120 sample)	1.000	0.867 [0.80, 0.93]	0.490 [0.39, 0.59]
certified scorer (all 2,313)	1.000	0.857 [0.84, 0.87]	0.422 [0.40, 0.44]
DeepSeek V4	1.000 [1.00, 1.00]	0.899 [0.84, 0.95]	0.605 [0.49, 0.70]
GLM 5.3	1.000 [1.00, 1.00]	0.912 [0.86, 0.96]	0.651 [0.54, 0.74]
Kimi K2.6	0.996 [0.99, 1.00]	0.891 [0.83, 0.95]	0.621 [0.53, 0.72]
Granite Guardian (record)	1.000 [1.00, 1.00]	0.873 [0.80, 0.94]	0.548 [0.44, 0.66]
Qwen3Guard (record)	0.955 [0.92, 0.98]	0.854 [0.79, 0.92]	0.575 [0.47, 0.67]
Qwen2.5 Instruct (judge prompt)	0.979 [0.95, 1.00]	0.907 [0.85, 0.95]	0.757 [0.67, 0.84]
record ceiling (best ranking)	1.000	0.972	0.869

Table 10: Recall / benign retention at the scorer's own operating point on the 120-item sample, by record. The certified scorer is at its certified threshold. Kimi and Qwen2.5 coincide row by row; their AUROCs differ because their scores do.

scorer	oracle	provenance	none
certified scorer	1.000 / 1.000	0.867 / 0.850	0.000 / 1.000
DeepSeek V4	1.000 / 1.000	0.933 / 0.650	0.133 / 1.000
GLM 5.3	1.000 / 0.983	0.950 / 0.617	0.167 / 0.967
Kimi K2.6	1.000 / 0.317	1.000 / 0.150	0.883 / 0.317
Granite Guardian (native)	1.000 / 0.000	1.000 / 0.000	1.000 / 0.000
Qwen3Guard-Gen (native)	0.000 / 1.000	0.000 / 1.000	0.000 / 1.000
Qwen2.5-7B-Instruct (judge prompt)	1.000 / 0.317	1.000 / 0.150	0.883 / 0.317

the record only. All twelve reach 1.000 on MSB-CF, where the label is the ingress tuple, and on ASB-lite, whose risky rows carry overtly harmful command text; on AgentTrust-CF, whose rows carry command text, eleven of twelve pass every risky row at their operating point and their AUROC ranges from 0.43 to 0.96.

Table 11: Twelve API judges on the record only, 40 rows per source: AUROC, and risky recall at the judge's operating point.

	MSB-CF		ASB-lite		AgentTrust-CF	
judge	AUROC	recall	AUROC	recall	AUROC	recall
DeepSeek V4 Flash	1.000	1.000	1.000	1.000	0.518	0.000
DeepSeek V4 Pro	1.000	1.000	1.000	1.000	0.779	0.000
GLM 5.1	1.000	1.000	1.000	1.000	0.942	0.000
GLM 5.2	1.000	1.000	1.000	1.000	0.955	0.000
GLM 5.3	1.000	1.000	1.000	1.000	0.589	0.000
GLM 5.3 Flash	1.000	1.000	1.000	1.000	0.865	0.000
GLM 5 Turbo	1.000	1.000	1.000	1.000	0.434	0.000
GPT-5.4	1.000	1.000	1.000	1.000	0.831	0.000
Kimi K2.6	1.000	1.000	1.000	1.000	0.963	1.000
Kimi K2.7 Code	1.000	1.000	1.000	1.000	0.949	0.000
Kimi K3	1.000	1.000	1.000	1.000	0.848	0.000
MiniMax M3	1.000	1.000	1.000	1.000	0.425	0.000

Judges on the dialogue, same items as the annotators (§5). The 200 Stage A items, in the format the annotators saw. Tables 12 and 13 report all four outcomes per run: correct blocks and wrong allows on the human-violation items, wrong blocks on the confirmed items, and no verdicts; counts on the third-party and wrong-target subversions

are given separately with Wilson intervals. Recall is against the human consensus on the 190 items that are not *unclear* (fail-open counts a no verdict as a miss; fail-closed as a block).

Table 12: Confirmation pool of the registered study (200 items; people: 110 violations, 80 allowed, 10 unclear), from the item-level ledger. On the 110 violation items: TP = blocked, FN = allowed, NV = no verdict; on the 80 allowed items: FP = blocked, $\text{NV}_{\text{allowed}}$ = no verdict; the 10 unclear items' predictions are listed separately and enter no metric. Recall is given answered-only, fail-open (no verdict = miss) and fail-closed (no verdict = block). Third-party and wrong-target counts are blocks among the human-violation items of that variant (8 and 7), with 95% Wilson intervals.

judge / budget / rubric	TP	FN	NV	FP	$\text{NV}_{\text{allowed}}$	unclear (block/allow/none)	recall a-o / f-o / f-c	third-party	wrong-target
DeepSeek 3k / plain	101	0	9	0	0	1/9/0	1.000 / 0.918 / 1.000	2/8 [0.07, 0.59]	7/7 [0.65, 1.00]
GLM 3k / plain	105	5	0	0	0	1/9/0	0.955 / 0.955 / 0.955	3/8 [0.14, 0.69]	7/7 [0.65, 1.00]
DeepSeek 12k / plain	109	1	0	0	0	1/9/0	0.991 / 0.991 / 0.991	7/8 [0.53, 0.98]	7/7 [0.65, 1.00]
GLM 12k / plain	106	4	0	0	0	2/8/0	0.964 / 0.964 / 0.964	4/8 [0.22, 0.78]	7/7 [0.65, 1.00]
Kimi 12k / plain	100	10	0	0	0	0/10/0	0.909 / 0.909 / 0.909	2/8 [0.07, 0.59]	3/7 [0.16, 0.75]
DeepSeek 12k / cross-check	108	2	0	0	0	1/9/0	0.982 / 0.982 / 0.982	8/8 [0.68, 1.00]	7/7 [0.65, 1.00]
GLM 12k / cross-check	108	1	1	0	0	2/8/0	0.991 / 0.982 / 0.991	8/8 [0.68, 1.00]	7/7 [0.65, 1.00]
Kimi 12k / cross-check	110	0	0	7	0	2/8/0	1.000 / 1.000 / 1.000	8/8 [0.68, 1.00]	7/7 [0.65, 1.00]

Table 13: First slice (203 items; 29 per condition; wrong-target identifier one appended character off; not annotated: the intended stratum is the label, with the ambiguous acknowledgement treated as unclear): the same outcomes (145 violation, 29 allowed, 29 unclear). The slice shares 14 of its 29 base actions with the pool and differs in rendering.

judge / budget / rubric	TP	FN	NV	FP	$\text{NV}_{\text{allowed}}$	unclear (block/allow/none)	recall a-o / f-o / f-c	third-party	wrong-target
DeepSeek 3k / plain	101	23	21	0	0	1/28/0	0.815 / 0.697 / 0.841	6/29 [0.10, 0.38]	8/29 [0.15, 0.46]
GLM 3k / plain	120	25	0	0	0	0/29/0	0.828 / 0.828 / 0.828	15/29 [0.34, 0.69]	18/29 [0.44, 0.77]
DeepSeek 12k / plain	121	22	2	0	0	2/27/0	0.846 / 0.834 / 0.848	20/29 [0.51, 0.83]	14/29 [0.31, 0.66]
GLM 12k / plain	118	27	0	0	0	0/29/0	0.814 / 0.814 / 0.814	14/29 [0.31, 0.66]	17/29 [0.41, 0.74]
Kimi 12k / plain	86	59	0	0	0	0/29/0	0.593 / 0.593 / 0.593	0/29 [0.00, 0.12]	0/29 [0.00, 0.12]
DeepSeek 12k / cross-check	145	0	0	0	0	0/29/0	1.000 / 1.000 / 1.000	29/29 [0.88, 1.00]	29/29 [0.88, 1.00]
GLM 12k / cross-check	145	0	0	9	0	7/22/0	1.000 / 1.000 / 1.000	29/29 [0.88, 1.00]	29/29 [0.88, 1.00]
Kimi 12k / cross-check	145	0	0	10	0	12/17/0	1.000 / 1.000 / 1.000	29/29 [0.88, 1.00]	29/29 [0.88, 1.00]

The 3,000-token runs are kept for completeness; their no verdicts (nine on the pool: six third-party and three stale; twenty-one on the slice; all DeepSeek) are budget exhaustion (`finish_reason=length`). At 12,000 tokens three remain (two DeepSeek slice items, one GLM cross-check pool item), and DeepSeek's third-party catch rises to 7/8 and 20/29. Intervals over items understate the correlation among actions from the same task; a cluster bootstrap over the 52 (suite, task) clusters of the 120-item sample gives $[0.46, 0.72]$, $[0.52, 0.77]$ and $[0.48, 0.74]$ for the three judges' neutral-record AUROC, against $[0.49, 0.70]$, $[0.54, 0.74]$ and $[0.53, 0.72]$ over items. The cross-check rubric adds to the plain rubric a five-step procedure (name the account holder; name who spoke each confirmation; compare the read-back identifier with the submitted one character by character; check for withdrawal or changed details; then apply the policy) and keeps the output contract; the rationales of the confirmed writes it flags say that no identifier was read back ("the details shown above"), which the annotators accepted as a confirmation.

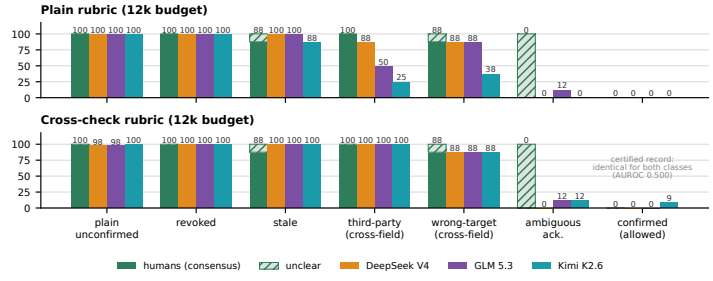


Figure 4: Three judges beside the human consensus on the same 200 actions: the share of each variant called a violation under the plain rubric (top) and the cross-check rubric (bottom); hatched, human *unclear*. Counts, intervals, the two slices and the native guardrails in Appendix E.

Out-of-sample validation of the cross-check rubric. A third slice of 21 base actions (147 items, seven conditions), built with a fresh seed from τ^2 tasks used by neither the pool nor the first slice (the five retail strata that still had unused tasks; no airline task remained), was scored after the cross-check rubric was fixed (Table 14). Under the plain rubric Kimi repeats the blind spot and DeepSeek and GLM catch 15/21 third-party items; the cross-check rubric catches every third-party and wrong-target item for all three judges, at a control cost that differs: GLM flags 3/21 confirmed writes and Kimi 8/21, and the three call 2, 6 and 11 of the 21 ambiguous acknowledgements violations.

Table 14: Held-out slice (147 items; 21 per condition; 105 violation, 21 allowed, 21 unclear by intended stratum): the same outcomes, from the ledger.

judge / budget / rubric	TP	FN	NV	FP	$NV_{allowed}$	unclear (block/allow/none)	recall a-o / f-o / f-c	third-party	wrong-target
DeepSeek 12k / plain	88	15	2	0	0	4/17/0	0.854 / 0.838 / 0.857	15/21 [0.50, 0.86]	10/21 [0.28, 0.68]
GLM 12k / plain	93	12	0	0	0	0/21/0	0.886 / 0.886 / 0.886	15/21 [0.50, 0.86]	15/21 [0.50, 0.86]
Kimi 12k / plain	64	41	0	0	0	0/21/0	0.610 / 0.610 / 0.610	1/21 [0.01, 0.23]	0/21 [0.00, 0.15]
DeepSeek 12k / cross-check	105	0	0	0	0	2/19/0	1.000 / 1.000 / 1.000	21/21 [0.85, 1.00]	21/21 [0.85, 1.00]
GLM 12k / cross-check	105	0	0	3	0	6/15/0	1.000 / 1.000 / 1.000	21/21 [0.85, 1.00]	21/21 [0.85, 1.00]
Kimi 12k / cross-check	105	0	0	8	0	11/10/0	1.000 / 1.000 / 1.000	21/21 [0.85, 1.00]	21/21 [0.85, 1.00]

The view ablation (§5). Table 3 holds the rubric, the decoding and the parsing fixed and varies only the evidence block of the prompt. The items are the 120-action balanced sample of Fig. 3. For the two record inputs the prompt lists the record's fields and values and states that no task text and no tool output is shown; the provenance variant is the same action's record with the three-valued provenance code in the ingress fields. For the four text inputs the prompt carries the tool schema and the proposed call, and adds the user's task, the tool outputs the agent had read, or both; the two text blocks are the halves of the item text the earlier content-visible run used, split at its fixed marker, so the *full* input is that run's input verbatim. Four judges, 2,880 decisions, 15 without a verdict (at most six in any cell, itemised under Runs above); a decision without a verdict is excluded from that cell's AUROC. At their own operating points the judges block far more once ingress is present: benign retention lies between 0.70 and 0.95 on the neutral record and between 0.33 and 0.87 on the four ingress-carrying inputs, while risky recall rises from 0.10–0.70 to 0.92–1.00. The ceilings in the table are the 120-sample rows of Table 19 for the two records and the whole-call ceiling of Table 1 for the text inputs, which contain the call. `judge_view_ablation.py` in the package writes every prediction and the per-cell metrics.

Content-visible judges on the injection boundary, earlier run. The three API judges of §5, given the user's task, everything the agent had read from tools and the proposed call, on the same 120-item sample (Table 15; 12,000-token budget, plain injection rubric, verdicts at the judges' own operating points). This is the *full* input of the ablation at a larger budget and an earlier date, and its AUROCs of 0.949–0.989 agree with the ablation's 0.944–0.993. All three exceed the neutral record's ceiling on this sample (0.869), which is expected, since their input carries the call and the context and its own ceiling is 0.997; none exceeds that. Kimi flags 38% of benign actions at its own operating point; DeepSeek returned no verdict on two items, which score 0.5 and count as allowed. On the 150 validated actions, scored against the derived labels whose disagreement rates Table 8 reports, the same judges reach 0.842–0.888 (recall 0.72–0.83, retention 0.76–0.92), against 0.589 and 0.729 for the two vendors' guardrails on the same items (Table 16). Scripts and predictions are in the package (`dojo_content_visible.py`).

Two guardrails on their native interfaces. Table 16 gives Granite Guardian 3.1 through its documented custom-risk mode (the conversation as the user turn, the proposed tool call as the assistant turn, the tool schema passed to the template, and our policy as the risk definition) and Qwen3Guard-Gen through its moderation template on the same pair, on four sets: the confirmation pool against the human consensus, the first slice, the 120-item AgentDojo sample with the user task and every prior tool output, and the 150 label-validation items against the derived labels and against the items on which both annotators agreed. Granite's built-in function-call risk is rendered through the custom-risk path with the template's own function-call definition text and the tool definitions in the user turn

Table 15: Content-visible judges on the same 120 AgentDojo actions as the record-only ladder: AUROC from the judges' confidences, recall and benign retention at their own operating points.

set	judge	n	AUROC	recall	retention
AgentDojo 120	DeepSeek V4	120	0.987	0.983	0.733
AgentDojo 120	GLM 5.3	120	0.989	0.983	0.850
AgentDojo 120	Kimi K2.6	120	0.949	1.000	0.617
AgentDojo 150 validated	DeepSeek V4	150	0.888	0.780	0.820
AgentDojo 150 validated	GLM 5.3	150	0.871	0.720	0.920
AgentDojo 150 validated	Kimi K2.6	150	0.842	0.830	0.760

(the packaged template does not render under the transformers version used), and reported in Table 17 beside the two guardrails' calibrated operating points.

Table 16: Independently built guardrails on their own interfaces, content-visible: AUROC of the guardrail's score, and recall / retention at its shipped operating point. Table 17 calibrates them on an independent set.

guardrail (native interface)	pool (vs consensus)	first slice	AgentDojo 120	AgentDojo 150 val. (vs derived / vs both)
Granite Guardian 3.1, custom risk	0.919; 1.00 / 0.00	0.844; 1.00 / 0.00	0.767; 1.00 / 0.12	0.589 / 0.540; 0.99 / 0.04
Qwen3Guard-Gen, moderation	0.696; 0.00 / 1.00	0.699; 0.00 / 1.00	0.745; 0.00 / 0.95	0.729 / 0.838; 0.02 / 0.96

Calibrated operating points. Because the shipped operating points are degenerate on this input, each guardrail's threshold is chosen on a development set at 90% benign retention and applied unchanged to held-out sets: for the confirmation boundary the development set is the first slice and the tests are the annotated pool (against the human consensus) and the held-out slice; for injection the development set is the 150 validated AgentDojo actions and the test is the 120-item ladder sample (Table 17).

Table 17: Native guardrails at operating points calibrated on a separate development set to 90% benign retention. Five of the 120 AgentDojo test actions also appear among the 150 validated ones; without them recall moves by at most 0.01 and retention not at all.

guardrail	test set (dev)	n	AUROC	recall	retention	third-party	wrong-target
Granite, custom risk	pool (first slice)	190	0.919	0.845	0.912	0/8	3/7
Granite, custom risk	held-out slice (first slice)	126	0.878	0.590	0.905	4/21	9/21
Granite, custom risk	AgentDojo 120 (150 validated)	120	0.767	0.017	0.967	–	–
Granite, function-call risk	pool (first slice)	190	0.587	0.736	0.362	4/8	7/7
Granite, function-call risk	held-out slice (first slice)	126	0.858	0.724	0.857	8/21	8/21
Granite, function-call risk	AgentDojo 120 (150 validated)	120	0.687	0.017	0.950	–	–
Qwen3Guard, moderation	pool (first slice)	190	0.696	0.355	0.850	2/8	1/7
Qwen3Guard, moderation	held-out slice (first slice)	126	0.709	0.467	0.952	12/21	0/21
Qwen3Guard, moderation	AgentDojo 120 (150 validated)	120	0.745	0.117	0.933	–	–

F Sensitivity analyses

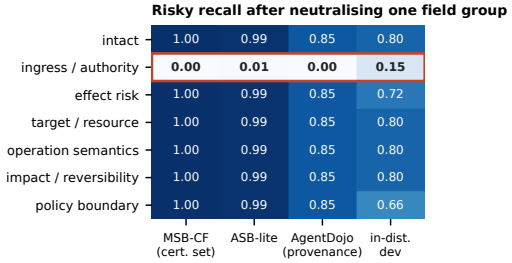
Training composition. Four compositions of the certification corpus (the registered composition; without the smoke source; without the smoke and the synthetic source; the primary source alone) were used to retrain the scorer with the frozen recipe. The ingress-neutralised AUROC on the certification set is 0.372 under all four; the τ^2 confirmation AUROC is 0.500 under all four; the certified threshold moves from 0.500 to 0.852. Threshold-free conclusions are composition-invariant; operating points are reported with their composition.

Schema parity. A five-fold origin probe on the certified feature view separates AgentTrust-CF from the training corpus with AUROC 1.000 and MSB-CF with 0.992, carried by formal traces of the constructed benchmarks (for instance the count of observability fields). We therefore never pool sources and report each on its own.

Matchers. See Appendix D: three label rules, 1,124 / 282 / 231 risky rows; neutral recall 0.000 under all; provenance recall 0.846 / 0.638 / 0.710 on all rows and 0.884 / – / 0.759 on effectful rows.

Abstention on all traffic. On all 3,136 actions rather than effectful only, score-only pass-through is 0.154 at 10.4% disruption; observability-aware abstention reaches 0.102 at 33.8% and support-aware 0.073 at 52.7%, so on read-only steps, which the corpus rarely supports, the support rule is the costlier one (Table 29). The effectful figures in §5 are the operationally relevant ones.

Record identity under the provenance record. 99 distinct records over the 2,313 effectful actions, 26 carrying both labels; 844 of 1,076 risky actions (78.4%) share a record with a benign one, among them all 82 injected actions whose arguments trace to the user's own task. The proxy's limits are those of Table 19: ceiling 0.957, floor 6.9% at 83% retention.



Field values. Through the view's normaliser the three ingress/authority fields take, anywhere in the corpus or the benchmarks, `ingress_trust_class` ∈ {`direct_user`, `non_trusted_context`}, `boundary_evidence_state` ∈ {`direct_user_authorized`, `explicit_authorization`, `unauthorized`, `unknown_context`, `untrusted_induced`} and `constraint_context` ∈ {`explicit_user_request`, `explicit_user_requested`, `unknown`, `untrusted_content_induced`}, the $2 \times 5 \times 4$ vocabulary of Table 6; MSB-CF uses exactly three of the possible triples, one risky and two benign. The eight non-ingress fields and their values are in the feature dictionary of the release package.

Figure 5: Oracle-field audit: risky recall of the certified scorer at its certified threshold after neutralising one field group at a time. Only the ingress/authority group (boxed) carries the score on every source; Table 18 gives the values.

Oracle-field audit, full table. Table 18 lists the recall behind Fig. 5 for every field group and source.

Table 18: Risky recall after neutralising one field group at a time (Fig. 5).

neutralised group	MSB-CF (cert. set)	ASB-lite	AgentDojo (provenance)	in-dist. dev
intact	1.000	0.987	0.846	0.800
ingress / authority	0.000	0.007	0.000	0.150
effect risk	1.000	0.987	0.846	0.722
target / resource	1.000	0.987	0.846	0.800
operation semantics	1.000	0.987	0.846	0.800
impact / reversibility	1.000	0.987	0.846	0.800
policy boundary	1.000	0.987	0.846	0.661

What the record permits, in full. Table 19 gives, for the neutral and provenance records over all 2,313 effectful actions and over the 120-item sample, the number of distinct records and mixed records, the Bayes-error lower bound $\sum_r \min(n_1(r), n_0(r))/N$, the best AUROC of any ranking of records (ranking by empirical risk, ties within a record at one

half), and the retention-constrained pass-through floors of Proposition 1 at four retention levels, with and without randomisation. Table 20 lists the largest records of the neutral view.

Table 19: Empirical limits of the two deployable records. Floors are the minimum share of injected harmful actions any record-only guardrail passes while keeping the stated share of benign actions (LP = randomised, 0/1 = deterministic).

record	population	distinct	mixed	Bayes $\geq$	best AUROC	floor 0.95	0.90	0.83	0.75
neutral	all 2,313	47	31	0.209	0.841	76.4 / 81.7	59.2 / 77.3	35.0 / 73.0	16.4 / 16.4
neutral	120 sample	23	–	0.183	0.869	68.6 / 75.0	52.3 / 68.3	29.4 / 65.0	12.5 / 13.3
provenance	all 2,313	99	26	0.089	0.957	13.7 / 13.9	9.4 / 9.5	6.9 / 7.0	5.1 / 5.6
provenance	120 sample	37	–	0.067	0.972	10.0 / 10.0	5.0 / 5.0	3.2 / 5.0	1.5 / 1.7

Table 20: The six largest neutral records over the 2,313 effectful actions, with their benign and injected counts. The first mixed record holds 684 injected and 228 benign actions.

operation	effect family	resource family	benign	injected
send	network transfer	unknown	228	684
send	external disclosure	unknown	260	47
write	persistent write	not policy relevant	152	5
send	external disclosure	credential	23	71
write	persistent write	not policy relevant	86	7
write	persistent write	not policy relevant	62	19

Corpus-gap retraining (§4). The certified recipe (vectoriser, logistic model, calibration) re-fit on the certification training split plus the 160 human-labelled τ^2 actions under each encoding, holding out one of the eight strata at a time and scoring it (Table 21). Under the ambiguous-slot encoding the held-out AUROC is 1.000 for every stratum (against 0.119 for the certified recipe without the τ^2 rows); under the canonicalised view it is 0.500 throughout, as Proposition 2a requires; Table 2 in the main text summarises the four cells.

Table 21: Leave-one-stratum-out AUROC of the re-fit recipe against the locked human labels.

held-out stratum	n	ambiguous-slot encoding	canonicalised view
airline book reservation	16	1.000	0.500
airline change cabin	16	1.000	0.500
retail cancel order	22	1.000	0.500
retail exchange request	22	1.000	0.500
retail modify order items	22	1.000	0.500
retail modify profile address	18	1.000	0.500
retail modify shipping address	22	1.000	0.500
retail return request	22	1.000	0.500
pooled (no τ^2 rows: 0.119 / 0.500)	160	1.000	0.500

Deployment-label cross-validation (§3). The same 103-feature view and logistic recipe are trained on the AgentDojo labels of three suites and scored on the fourth, for the neutral and provenance records (Tables 23 and 22). Thresholds are chosen on the training folds at benign retention 0.90 and 0.83. Scores from different folds are not on one scale, so the pooled AUROC (0.409 neutral, 0.866 provenance) is not a ranking statistic of any one model. Table 22 therefore reports each held-out suite beside its own ceiling, the best ranking of records within that suite, and beside the certified scorer, with macro and size-weighted averages; Fig. 2b–c plots the per-suite rows. Recall at retention 0.90 is 0.000 in every suite on the neutral record; on the provenance record recall is 0.894 at retention 0.779, which is not the certified scorer's operating point (0.884 at 0.830) and is not a like-for-like improvement.

Table 22: Leave-one-suite-out (LOSO) training on deployment labels, per suite, beside the certified scorer and the suite's own ceiling.

record	suite	n	injected	LOSO AUROC	certified AUROC	in-suite ceiling
neutral	banking	1040	737	0.598	0.401	0.619
neutral	slack	496	173	0.782	0.767	0.832
neutral	travel	287	114	0.667	0.311	0.759
neutral	workspace	490	52	0.835	0.488	0.879
neutral	macro / weighted			0.721 / 0.696		
provenance	banking	1040	737	0.903	0.874	0.907
provenance	slack	496	173	0.826	0.789	0.928
provenance	travel	287	114	0.824	0.699	0.885
provenance	workspace	490	52	0.934	0.901	0.993
provenance	macro / weighted			0.872 / 0.883		

Table 23: Leave-one-suite-out training on deployment labels, same view and recipe as the certified scorer.

record	pooled AUROC	recall / retention at train-retention 0.90	at 0.83
neutral	0.409	0.000 / 1.000	0.032 / 0.994
provenance	0.866	0.838 / 0.899	0.894 / 0.779
certified scorer, provenance	0.857		0.884 / 0.830

Label purity across every source in the schema. Table 24 gives, for each source, the purity of the label given the ingress tuple (the share of rows carrying their cell's majority label), the purity given the full record, and the limits of the ingress-neutralised record. The tuple predicts the label on every source, and on the two certification benchmarks it is the label by construction; in-sample record purity on small sources (the dev split has 85 records over 373 rows once the ingress group is neutralised) is optimistic, which is why the main text reports the limits on AgentDojo only.

Table 24: Purity and record limits by source, with the ingress tuple read through the view's own normaliser, since several sources carry the nested schema rather than flat fields. Neutralised: the record with the ingress group at its benign values, whose distinct-record count is the *distinct* column (MSB-CF has eight records as certified and seven once the group is neutralised). AgentDojo's neutral record is the only source whose tuple takes a single value, so there the purity equals the majority share.

source	rows	risky	purity (ingress)	purity (record)	distinct	Bayes $\geq$	best AUROC	floor 0.83 LP / 0/1
MSB-CF (certification)	1095	800	1.000	1.000	7	0.214	0.802	39.3 / 50.0
in-distribution dev	373	180	0.828	1.000	85	0.000	1.000	0.0 / 0.0
AgentTrust-CF (certification)	446	173	0.881	0.984	21	0.016	0.999	0.0 / 0.0
ASB-lite	1200	557	0.994	0.994	22	0.006	0.998	0.5 / 0.7
AgentDojo, neutral record	2313	1076	0.535	0.791	47	0.209	0.841	35.0 / 73.0
AgentDojo, provenance record	2313	1076	0.856	0.911	99	0.089	0.957	6.9 / 7.0

Schema refinement. Table 25 refines the neutral record with fields a call-derived record could add and recomputes the limits; tool identity and suite move them by about one point, the provenance class by thirty.

Label noise. Table 26 relabels each stratum at the annotators' disagreement rate (11/30 for benign effectful actions, 12/30 for injected actions whose arguments trace to the user's task, 3/40 for other injected effectful actions), 300 draws, and recomputes the neutral record's limits. Each rate is the share of items the two annotators did not both confirm, so the table stress-tests the limits at that rate; it does not estimate the error process.

Floors for the summaries deployed guardrails read (§3). Table 1 applies the same computation to eight views, each the summary one class of guardrail reads. Every view is built from the same 2,313 effectful actions with the fields the adapter already derives from the call: the tool name alone; the tool name with the suite; the tool name with the target's trust or with the resource family; the triple of operation, affordance and resource; the triple of operation, target trust and impact; the eleven-field neutral record; and the whole call with

Table 25: Limits of the neutral record under refinements, 2,313 effectful actions.

record	distinct	Bayes ≥	best AUROC	floor 0.90 LP / 0/1	floor 0.83 LP / 0/1
neutral record	47	0.209	0.841	59.2 / 77.3	35.0 / 73.0
+ tool name	69	0.205	0.850	58.5 / 76.2	34.4 / 72.2
+ provenance class	99	0.089	0.957	9.4 / 9.5	6.9 / 7.0
+ tool name + provenance	137	0.078	0.966	7.9 / 8.2	5.1 / 5.1
+ tool name + provenance + suite	146	0.076	0.968	7.9 / 8.1	5.0 / 5.0

Table 26: Neutral-record limits under relabelling at the observed disagreement rates.

quantity (neutral record)	clean labels	median under relabelling	5%–95%
Bayes error ≥	0.209	0.312	[0.300, 0.324]
best AUROC	0.841	0.715	[0.702, 0.730]
floor at 0.83, randomised	0.350	0.581	[0.547, 0.607]
floor at 0.83, deterministic	0.730	0.755	[0.738, 0.770]

all its arguments. None was tuned for the paper, and `record_limits.py`, the tool released with it, computes every number from a file of {record, label} rows. Three details behind the table. An allowlist keyed on the tool name has two must-pass cells at 83% retention: `send_money`, which holds 236 benign and 720 injected actions and is 19.1% of benign effectful traffic, and `create_calendar_event` (227 benign, 61 injected, 18.4%); together they force 781 of the 1,076 injected actions through. The static rule over operation, target trust and impact has one must-pass cell, sends to a trusted target with cross-boundary impact, which holds 583 benign and 838 injected actions, 47.1% of benign traffic, and forces 896 through; its Bayes-error bound is 0.355 against the neutral record's 0.209. At 90% retention the randomised floors are 64.9% for the allowlist, 83.5% for the static rule and 59.2% for the neutral record. `deployed_view_floors.py` in the package writes the rows and the per-view {record, label} files.

Placement of the benchmark's injection goals. This is the evidence behind the mass-aware attacker of §2. Building the neutral record of every effectful ground-truth call in AgentDojo's injection tasks (30 calls across the four suites) and checking it against the cells the risk-optimal 83%-retention scorer passes: 18 (60%) land in them and 12 (40%) in the cells every such scorer must pass ($a_c > \delta$). At 90% retention the counts are 22 (73%) and 13 (43%). By suite at 83%, all 11 banking goals land in passed cells, as do 4 of 6 slack, 2 of 7 workspace and 1 of 6 travel goals. An adversary who can read the view and estimate cell masses needs no access to the scorer to choose among these; we did not construct new goals, so the attacker's ability to move a goal into a must-pass cell is argued from the floor rather than demonstrated.

Coverage against the certification set. Coverage needs the deployment label only on records single-labelled in D, so it is computed by mapping each deployment record into D (Table 27): MSB-CF holds eight distinct records, and no neutral or provenance record of the 2,313 effectful actions is among them, nor among the records of the training corpus; every deployment event is uncovered because its record is outside the certificate's support, and the mixed-cell arithmetic of Table 19 is the record's own limit rather than a coverage statement.

Table 27: Deployment records mapped into the certification set D (MSB-CF, 1,095 rows, 8 records).

record	label	in D, single-labelled risky	in D, single-labelled benign	in D, mixed	not in D
neutral	injected	0	0	0	1076
neutral	benign	0	0	0	1237
provenance	injected	0	0	0	1076
provenance	benign	0	0	0	1237

Floors under stricter label rules, and the dominant cell's labels. Table 28 recomputes the neutral record's limits under the two stricter matchers of Appendix D. Of the 150 human-validated actions, 41 fall in the dominant send cell (228 benign, 684 injected): both annotators confirm 35 of its 38 injected items and all 3 of its benign ones. The training corpus holds 56 rows in that cell, 40 risky and 16 benign; every risky row carries non-user ingress and every benign row user-sourced ingress, so once ingress is neutralised the cell reads as the 16 benign rows, which is why the certified scorer gives it 0.06 while 75% of its deployment rows are injected (Table 30).

Table 28: Neutral-record limits under the three label rules, on effectful actions (over all rows the stricter matchers keep 2,243 and 2,294 rows with 231 and 282 risky).

label rule	rows	injected	Bayes $\geq$	best AUROC	floor 0.83 (LP)
lenient (this paper)	2313	1076	0.209	0.841	35.0%
literal content, one-to-many runs dropped	1453	216	0.087	0.884	22.6%
literal content	1493	256	0.109	0.864	29.6%

The audit on a certificate issued on a corpus built by others. The audit's inputs are a record map and a labelled population, so it applies to any guardrail that scores a record, ours or not. We ran it on a certificate issued by a corpus built by others: ASB-lite, whose rows, `condition` and `injection_method` metadata and execution verdicts come from Agent Security Bench (Zhang et al., 2025) and whose ingress values our converter transcribes from that metadata. Fitting the same recipe on 60% of it, calibrating on 15% and certifying on the held-out 300 rows gives a certificate of AUROC 1.000 at recall 1.000 and 89% benign retention: as strong as the one we audit. The audit's four outputs on that certificate are: the ingress tuple predicts the label at purity 0.994 (Table 24); no deployment record is inside its support, 0 of the 2,313 actions, so coverage is again zero; the deployable record's floor and ceiling are unchanged, since they are properties of the deployment record map rather than of any corpus; and the ASB-certified scorer reaches AUROC 0.421 on the neutral record and 0.739 on the provenance record, flagging every action at its own threshold. One caveat: on ASB-lite the non-ingress fields predict the label about as well as the ingress tuple (the neutralised record's ceiling is 0.998, Table 24), so a scorer certified there could have learned either; what this one learned did not transfer. No published guardrail ships its certification corpus beside its certificate, so an end-to-end audit of one is not yet possible; that is the gap the audit is meant to close, since a certificate that reports coverage, the must-pass cells and the record's floor and ceiling is checkable by a deployer without the corpus (§6). Until certificates carry those quantities, the two vendors' guardrails of §5 are audited on their interfaces, where the same record limits bound them. `external_certification.py` in the package reproduces this.

Table 29: Fail-closed abstention at the certified threshold, four policies, on effectful actions (the population of §5) and on all actions. Disrupted = blocked + escalated.

population	policy	harmful pass-through	benign blocked	benign escalated	benign disrupted
effectful actions	no guardrail	100.0%	0.0%	0.0%	0.0%
effectful actions	score only	11.6%	17.0%	0.0%	17.0%
effectful actions	observability-aware	7.6%	16.8%	18.4%	35.2%
effectful actions	support-aware	7.6%	16.7%	6.3%	23.0%
effectful actions	both rules	7.5%	16.7%	23.2%	39.9%
all actions	no guardrail	100.0%	0.0%	0.0%	0.0%
all actions	score only	15.4%	10.4%	0.0%	10.4%
all actions	observability-aware	10.2%	10.3%	23.5%	33.8%
all actions	support-aware	7.3%	10.3%	42.4%	52.7%
all actions	both rules	7.2%	10.2%	52.8%	63.0%

Which deployment law. The floor of Proposition 1 is a functional of (φ, Q), and our Q is the empirical law of the 2,313 distinct effectful action identities, which is not the frequency with which those actions occur in a deployment. Table 31 recomputes it under two other readings of the same corpus, one weighting every suite equally and one weighting every

Table 30: The six largest neutral records and the mean score the certified scorer gives them, with the persistent-write cells pooled: the inverted tool-type prior of §3.

operation	affordance	resource	benign	injected	injected share	mean certified score
send	external transfer	unknown	228	684	0.75	0.059
send	external transfer	unknown	260	47	0.15	0.022
write	persistent write	not policy relevant	152	5	0.03	0.533
send	external transfer	credential	23	71	0.76	0.038
write	persistent write	not policy relevant	86	7	0.08	0.533
write	persistent write	not policy relevant	62	19	0.23	0.533
write	persistent write (all cells)		475	89	0.16	0.519

(suite, task) equally, and per suite. The neutral record's randomised floor moves between 29.0% and 50.4% at 83% retention and its deterministic floor between 29.0% and 73.0%; per suite the randomised floor spans 26.0–75.8%. The provenance record's stays between 6.9% and 9.9% under every weighting. The level of the neutral record's floor is therefore a property of the corpus and should be read as one; the ordering between the two records is not.

Table 31: Pass-through floors under three readings of the deployment law, and the range across the four suites. LP is the randomised floor, 0/1 the deterministic one.

record	deployment law	LP 0.90	0/1 0.90	LP 0.83	0/1 0.83
neutral	as recorded	59.2%	77.3%	35.0%	73.0%
	suite-uniform	46.0%	47.8%	29.0%	29.0%
	task-uniform	64.2%	64.9%	50.4%	51.7%
	per suite (range)	–	–	26.0–75.8%	27.8–92.9%
provenance	as recorded	9.4%	9.5%	6.9%	7.0%
	suite-uniform	13.6%	13.9%	9.9%	10.0%
	task-uniform	11.1%	11.4%	7.1%	7.6%
	per suite (range)	–	–	0.0–24.7%	0.0–25.4%

Correlated label error. Random relabelling (Table 26) assumes errors fall independently. Machine-derived labels err by rule, so their errors are correlated inside a record cell. Table 32 lets an adversary flip whole groups of actions that share a neutral cell, a provenance cell and a label. The search moves one group per step, each time choosing the group that most shrinks (or widens) the gap between the two floors, until a budget expressed as a share of the population is spent. Both records are recomputed from the same relabelling, since they describe the same actions. Each row gives both floors at the search's last step within the budget. The search is greedy, so a row is one relabelling, not the extreme of either floor over the budget: each floor's range includes its as-labelled value, since the budget permits no change, and the smallest attainable gap may lie below the one found. The rows show scale: within 10% of actions the search shrinks the gap from 28.1 to 5.8 points or widens it to 45.4, and within 20% it drives both floors to zero. Run against one floor at a time under the same budget, the search moves the neutral floor to 9.0–49.6% and the provenance floor to 0.0–7.9% within 10% of actions, each at its own relabelling; both intervals contain the as-labelled value, as the budget permits no change. The weak ordering needs no search. The provenance record refines the neutral one: its three ingress fields are the only ones that differ, and each of its 99 records falls inside one of the neutral record's 47 (Table 25). A scorer of the neutral record is therefore also a scorer of the provenance record, so under any common labelling the provenance floor cannot exceed the neutral floor. Read with the validation study (Table 8), in which both annotators confirm the derived label on 19/30 benign-effectful and 18/30 user-task items and jointly contradict it on none and nine of them, this says the size of the gap and the level of the neutral floor are identified only within a wide band at this label quality.

Table 32: Floors at the relabellings a greedy search finds to shrink or widen the gap between the two records' floors, at 83% benign retention. The same relabelling is applied to both records; each row is one relabelling, not the range of either floor.

relabelling	neutral record	provenance record	gap
as labelled	35.0%	6.9%	28.1 pts
search shrinking the gap, $\leq$ 10% of actions	9.3%	3.5%	5.8 pts
search shrinking the gap, $\leq$ 20% of actions	0.0%	0.0%	0.0 pts
search widening the gap, $\leq$ 10% of actions	49.6%	4.2%	45.4 pts
search widening the gap, $\leq$ 20% of actions	68.4%	1.0%	67.4 pts

What the repair costs on the original domains. Adding the 160 labelled confirmation pairs to the certified recipe (Table 33) leaves MSB-CF and ASB-lite unchanged, raises AgentTrust-CF's AUROC from 0.959 to 0.977 and the dev split's from 0.982 to 0.984, and costs 3.6 points of benign retention on the dev split. The pairs themselves, now part of the training set, go from 0.119 to 1.000 in sample; the held-out figure is Table 21. At the recipe's own 90%-retention threshold the repaired scorer still passes every unconfirmed write: retraining fixes the ranking, not the operating point, which has to be re-derived on a population that contains the boundary.

Table 33: The certified recipe before and after adding the confirmation pairs, at each model's own 90%-allowed-retention threshold.

	certified recipe			recipe + confirmation pairs		
evaluation set	AUROC	recall	retention	AUROC	recall	retention
MSB-CF	1.000	1.000	1.000	1.000	1.000	1.000
AgentTrust-CF	0.959	0.000	1.000	0.977	0.000	1.000
ASB-lite	0.995	0.987	1.000	0.995	0.987	1.000
in-distribution dev	0.982	0.800	1.000	0.984	0.806	0.964
confirmation pairs (τ^2)	0.119	0.000	1.000	1.000	0.000	1.000