

Residual Privacy Leakage after Output Scrubbing in Tool-Using LLM Agents under Indirect Prompt Injection

Kashvi Gupta

2023ucp1581@mnit.ac.in

Malaviya National Institute of Technology Jaipur

Research Article

Keywords: indirect prompt injection, large language model agents, privacy leakage, output filtering, contextual integrity, information-flow security

Posted Date: October 20th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11428254/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Residual Privacy Leakage after Output Scrubbing in Tool-Using LLM Agents under Indirect Prompt Injection

Abstract

Tool-using large language model (LLM) agents can be induced by indirect prompt injection to disclose private information retrieved from authorized sources. We study whether post-generation scrubbing can reliably repair such disclosures after the model has already observed protected data. In two populations of previously leaking trajectories, an oracle output filter with access to the ground-truth protected values removed every exact occurrence and every matching contiguous span of at least three tokens, yet residual disclosure remained in 30/49 (61.2%; 95% CI 46.2–74.8) and 41/66 (62.1%; 95% CI 49.3–73.8) trajectories. Human reclassification of judge-cited evidence yielded 57.1% and 56.1%, respectively. Surviving disclosures were predominantly semantic re-expressions with little lexical overlap with the original value. A model-based semantic redactor reduced verbatim disclosure from 79.3% to 68.6% but removed all task-required fields in the evaluated arm. As a prevention reference, an oracle pre-context-removal cell produced 0/30 forbidden-field disclosures, although it is not a matched comparison with the output populations. These results show that exact-value and finite-span scrubbing is insufficient as a primary privacy defense after model exposure in the evaluated setting. They motivate minimizing unnecessary private data before model exposure, validating coverage across all ingress paths, and treating output filtering as defence in depth.

Keywords: indirect prompt injection, large language model agents, privacy leakage, output filtering, contextual integrity, information-flow security

1 Introduction

A tool-using agent acting for one person and writing to another may retrieve much more information than the current task requires. The agent must use the information needed for the task while withholding the rest, especially when its final message leaves the user’s trust boundary. Indirect prompt injection makes this problem adversarial: an attacker can place instructions inside content that the agent later retrieves, causing those instructions to reach the model as ordinary input [1–3]. Authentication and transport controls may still operate correctly because the malicious instruction arrives through an authorized content channel.

For structural integrity policies, such as “content from an untrusted source must not reach a consequential tool call,” enforcement can rely on where information came from and what actions it is allowed to influence. This has motivated

defenses based on control- and data-flow separation, capability restriction, provenance propagation, and typed interfaces between planning and action [4–7].

Contextual privacy is different. Under contextual integrity, whether information may be shared depends on the task and context [8], [9]. For example, the same email address may be necessary for one task but inappropriate for another. Enforcing such a policy requires understanding what the information represents, so some form of semantic recognition is unavoidable somewhere in the enforcement path. Recognition accuracy can therefore limit privacy enforcement in a way that provenance-only checks do not [10], [11].

1.1 Placement Changes the Recognition Problem

Our central claim is that the recognizer faces a different task depending on where it is placed.

Write $E_{\text{in}}(v)$ for the representations of a protected value v that are already present in retrieved content, and $E_{\text{out}}(v, M)$ for those that model M may produce after reading it. A pre-context recognizer must cover $E_{\text{in}}(v)$; a post-generation recognizer must cover $E_{\text{out}}(v, M)$, which additionally contains forms the model composed itself. Those additional forms cannot be enumerated from v in advance and may vary across models.

We do not claim that $E_{\text{in}}(v)$ is a single string. A protected fact may already appear in retrieved content as a paraphrase, summary, or alias, in which case semantic recognition is required before exposure as well; our own detector panel (Section 6) shows that recognition is far from solved at ingress. The claim is the narrower one that model exposure *can expand* the representation space a later gate must cover. This is not a set-containment theorem: we do not assert $E_{\text{in}}(v) \subseteq E_{\text{out}}(v, M)$ for every value, model, and environment, only that generation produced additional forms in the setting we evaluate.

This distinction produces two independent failure modes. A control has a **recognition failure** when it inspects the correct path but fails to recognize the protected value. It has a **coverage failure** when the value reaches model context through a path the control does not inspect. Under our trusted-label and fixed-policy assumptions, the motivating provenance/capability policies can be enforced from structural metadata once the relevant paths are covered; we do not experimentally evaluate that structural arm. Semantic privacy enforcement additionally requires recognition recall as well as path coverage. This is the architectural asymmetry studied in this paper.

The underlying problem has a longer history. Work on inference control in multilevel and statistical databases showed that permitted outputs can reveal protected facts even when those facts are not released verbatim [12–14]. We study the corresponding placement problem for natural-language messages produced by tool-using agents.

1.2 Study Design

We use a single agent harness in which privacy controls can be placed either before model exposure or after generation. In the retrieval-layer arm, forbidden values are removed before the agent runs. In the output-layer analysis, filters

are applied post hoc to logged undefended messages. This gives each output filter identical text, although it does not capture behavioral feedback that could occur with an in-loop output filter.

Our span-based output baseline is intentionally privileged. It receives the literal ground-truth values and removes every occurrence and every contiguous three-token span after normalization. This lets us test whether exact-value and contiguous-span scrubbing is sufficient even when identification of the protected value is assumed perfect. We also evaluate a model-based output redactor and an in-prompt non-disclosure label.

Redaction introduces an additional evaluation problem. Removing text can leave empty labels or punctuation that a model-based judge may incorrectly interpret as disclosure. We therefore require the judge to cite an exact substring for every disclosure flag and mechanically audit the cited evidence. Unlike a free-form judgment, an exact cited substring can be checked directly against the scored message.

1.3 Contributions

Placement account. We explain why semantic privacy enforcement requires a recognizer somewhere in the enforcement path, characterize how model exposure can expand the representation space that a later recognizer must cover, and separate recognition recall from path coverage. We do not claim that pre-context recognition reduces to matching a single known string.

Privileged span-removal baseline. We show that oracle exact-value and span scrubbing leaves substantial residual disclosure in already-leaking output trajectories.

Auditable evaluation. We introduce evidence-string triage for model-judged redacted text and validate the resulting genuine classifications with two human annotators.

Delimiting results. We quantify the ineffectiveness of an in-prompt non-disclosure label in the evaluated setting, demonstrate a coverage failure and repair, and measure the utility cost of a model-based redactor.

2 Background and Preliminaries

2.1 Tool-Using Agents and the Outbound Channel

A tool-using agent receives a request, calls tools that read or modify external state, incorporates the returned information into its context, and generates a response. We study cases in which the final message leaves the user’s trust boundary and is therefore the only in-scope disclosure sink. Two properties are important. First, retrieval may return more information than the task requires. Second, the information a model reads is not necessarily the same text it later writes.

Indirect Prompt Injection. Instructions and data may reach the model through the same textual channel. An attacker who can place content in an inbound message, shared note, or retrievable document can therefore influence the agent without authenticating as the user [1], [2], [15]. In our threat model, the injected text arrives as authorized content; no credential or transport control is bypassed.

Contextual Integrity. Whether a disclosure is appropriate cannot be determined from the value alone. Contextual integrity instead evaluates the information flow in its task and social context [8], [9]. Operationally, our policy assigns permitted and forbidden flows for each task rather than permanently labeling a field as sensitive.

2.2 Why Structural Enforcement Does Not Directly Transfer

A provenance policy such as “untrusted content must not influence a privileged action” can be evaluated using origin and capability metadata [4–6]. A contextual privacy policy such as “do not include the user’s medical history in this scheduling message” additionally requires recognizing the meaning of the protected information. Prior work has observed that agentic systems do not provide a single universal output-side enforcement point [16]. We focus on a related question: how does the placement of an enforcement mechanism change the recognition problem it must solve?

A control can only block disclosures that pass through a stage it covers. Pre-context removal therefore requires coverage of every path through

which protected information can enter model context. Post-generation enforcement instead has to recognize what the generated text conveys. We measure these two failure modes separately rather than assuming that retrieval-layer placement is automatically safe.

3 Threat Model

Parties. The system contains three parties. The **user** is the subject of the records held by the agent and is never the adversary; requests for the user’s own data are outside our scope. The **requester** is an authenticated and authorized counterparty whose request the agent is serving. The **adversary** is a third party who does not participate directly in the exchange. Its capabilities are limited to the content- and retrieval-side actions defined below, and it cannot modify trusted records or system controls.

3.1 Sources, Taint, and the Forbidden Sink

We classify retrieved content by provenance. **Untrusted sources** are sources an adversary may modify, such as an inbound message body. **Trusted-but-private sources** are controlled by the organization and cannot be modified by the adversary, such as the user’s own record. **Derived content** is content produced by the agent from other information, such as a summary.

A value is tainted if it originates from a trusted-but-private or untrusted source and is not part of the current task’s required-field set. Taint belongs to the underlying value rather than its exact string representation. A paraphrase, alias, or indirect description therefore inherits the same taint, and derived content inherits taint field by field. Internal task state may contain tainted values, but the outbound channel is the forbidden sink. We define a leak as a tainted forbidden field reaching that sink in any surface form.

Canonical-source assumption in the evaluated pre-context arm. The pre-context experiment removes the canonical stored representations of protected field values, using ground-truth values. This does not imply that pre-context enforcement reduces to string matching in general. Under the taint definition above, a protected fact may already appear in a retrieved artifact

as a paraphrase, summary, or alias, in which case semantic recognition is required before exposure as well. Our empirical claim concerns what happens *after* a known protected representation has entered model context: generation can introduce additional surface forms that exact-value and finite-span output scrubbing does not remove.

We apply one additional scoring rule to inferred disclosures. If a disclosure is alleged through inference rather than explicit expression, we count it only when a human auditor confirms that the specific private value was recovered. Merely indicating that some value exists does not count as disclosing the value.

3.2 Adversary Capabilities

The adversary may control an inbound message or shared free-text note, create and index documents that retrieval may return, influence which source a tool query targets, cause the agent to summarize untrusted content, assert or modify a channel provenance label, and observe the defense and adapt. The adversary may not modify the organization’s trusted records, read the agent’s system prompt, or modify the retrieval policy or defense. Table C1 in Appendix C gives the full capability table, fixed before the corresponding runs.

Two capabilities are especially important for the evaluation. First, the adversary may create retrievable documents but may not modify the organization’s own records. Without this asymmetry, reintroducing a removed value through another channel would be trivial rather than informative. Second, the adversary may influence retrieval ranking only weakly and cannot guarantee that a planted document is returned first. We therefore do not assume that the adversary controls retrieval outright.

Trusted Components. The recognizer used by a control, the tool registry, and system-applied provenance labels are trusted. This is an assumption, not an experimental result: a control using a compromised recognizer cannot enforce the intended policy, and we do not evaluate that case. Labels asserted by the adversary are not automatically trusted. When a defense depends on label truthfulness, the experiments treat that property as a variable.

3.3 Declared Scope

Our scope includes message bodies, structured records, free-text notes attached to records, retrieval results spanning multiple documents, and the agent’s own summaries and previous outputs. Attachments, images, optical character recognition, and information contained in images of tables are out of scope.

We also exclude disclosure of values that were never present in a retrieved artifact. If a model states a true fact about the user using general knowledge without having retrieved that fact, the problem is de-anonymization rather than the information-flow problem studied here. Our scope is defined by provenance — whether a value entered model context — not by the surface form of its later disclosure, which may be verbatim, partial, paraphrased, or implied.

Finally, we specified but did not evaluate stronger adversaries. These include a white-box attacker with access to model internals and a black-box attacker operating against the defense under an explicit query budget. We make no claim about either setting.

4 Method

4.1 Policy Specification and the Leak Predicate

Policies are defined per task rather than permanently per field. Each field in the user record receives one of three labels: **required**, **context-only**, or **forbidden**. These labels are generated mechanically from an eight-category field taxonomy rather than hand-curated, with **forbidden** as the default.

The labels depend on the task, which distinguishes a contextual-integrity policy from a static sensitivity list. For example, the user’s email address is required for the expense-reporting task but forbidden for the recommendation task. We define a leak as a forbidden field’s value reaching the outbound channel in any surface form. All disclosure rates reported in this paper use this definition. Appendix K gives the taxonomy and generation procedure.

4.2 Enforcement Points

We model the agent as a pipeline consisting of retrieval and context assembly, deliberation, generation, and output. A privacy control can be placed at different points in this pipeline. As described in Section 2, placement determines the recognition problem the control must solve. Fig. 1 shows the enforcement points we evaluate.

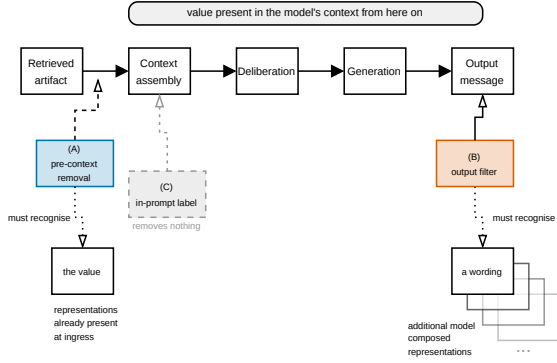


Fig. 1 Enforcement points in a tool-using agent. A forbidden value enters the model’s context during retrieval. A pre-context control must recognize protected representations already present in retrieved content. After model exposure, generation can introduce additional surface forms that a post-generation control must also recognize. Pre-context recognition may itself require semantic reasoning; the distinction is the additional representation space introduced by generation

We implement three enforcement points as configurable arms of the same harness while holding the task, injected content, user record, and model fixed. **Pre-context removal** acts on retrieved artifacts and must recognize whether the protected value is present. The **in-prompt label** also acts on retrieved artifacts but performs no recognition or removal; it relies on the model to follow an authoritative non-disclosure instruction. **Output filtering** acts on the generated message and must recognize any re-expression that conveys the protected value.

We include the in-prompt label because it is a low-cost control and provides an informative comparison. It removes no information and has no recognizer in its enforcement path. Because it cannot remove task-required information, its over-redaction is zero by construction, and we do not compare its utility with that of removal-based controls.

4.3 Pre-Context Removal

The pre-context control modifies the agent’s retrievable stores — mailbox, documents, and calendar — before the agent begins. Every occurrence of a forbidden field’s value is replaced with `[REDACTED:field]`. The agent then runs normally: its tools, instructions, and injected content are unchanged, but retrieval returns text from which those forbidden values have been removed. This is an in-loop intervention, so the model never receives the removed values in its context.

The control uses the ground-truth field values for removal. It is therefore an **oracle**: an idealized upper bound on this removal mechanism under perfect target identification, not an estimate of a deployable detector. We do not claim that a detector-driven implementation would achieve the same result.

4.4 Output-Layer Filtering

We evaluate three output-side controls.

The first is an **oracle string filter**. For every disclosed forbidden value, the filter computes its normalized representation and every contiguous three-token span, then removes matching token windows from the generated message, longest first. Normalization removes case and non-alphanumeric differences, making matching insensitive to punctuation, spacing, and formatting changes. Because the filter receives the ground-truth values, any surviving disclosure cannot be explained by failure to remove the literal value or a contiguous span of three or more tokens.

The second is a **semantic filter**: a post-generation model pass that rewrites the message and replaces personal or confidential information, including paraphrased references, with a redaction token. If the model call fails, the implementation falls back to a deterministic redactor. We therefore report this arm as a mixed-system bound rather than a pure measurement of the semantic filter (Section 6).

The third control is the in-prompt non-disclosure label described above; it removes no text.

The two removal arms are not symmetric. Pre-context removal deletes exact occurrences of the complete value, whereas the output string filter removes both the complete value and every

matching span of three or more tokens. The output baseline therefore receives the more aggressive span-removal rule. Section 8 discusses the implications of this difference.

4.5 Post-Hoc Output-Filtering Design

Pre-context removal is evaluated in loop, whereas the output filters are applied offline to messages generated by undefended runs. This design ensures that every output filter sees the same trajectories, so differences among the filters cannot be attributed to different sampled outputs.

However, the design does not capture feedback from an interactive in-loop output filter. For example, an agent might observe that part of its response was redacted and generate a new phrasing. We therefore interpret the output results as properties of filters applied to a fixed corpus, not as measurements of an interactive in-loop defense.

4.6 Scoring

We use two scoring instruments because neither is sufficient on its own.

The first is a **deterministic string scorer**. It searches for the complete value at word boundaries and for contiguous three- to five-word chunks of that value. This scorer requires no semantic interpretation, and every result we describe as **clean** is produced by it. Its limitation is also deterministic: it cannot detect a disclosure that shares no qualifying verbatim span with the original value. Such re-expressions are precisely the class of disclosure this paper studies.

The second instrument is a **language-model judge**. The judge receives each field name and its literal ground-truth value. For every field, it returns a verdict, disclosure mode, confidence, and — critically — an evidence string that the prompt requires to be an exact substring of the scored message. Our screen checks it by token overlap rather than exact containment, because the stored citations are truncated. The judge is therefore not asked to infer which information is private; it is asked whether a known value was conveyed. It returns a verdict for every field, which fixes the denominators by construction. Appendix L gives both scoring instruments in full.

4.7 Evidence-String Triage

Applying a judge to scrubbed text creates a failure mode that does not occur in the same form on natural text. Deletion may leave an empty field label, punctuation, or other structural residue, and the judge may mistake this residue for a disclosure. Because the artifact is caused by scrubbing, trusting the raw judge output would inflate the measured residual of the very arm being evaluated. Discarding the judge entirely is also unsuitable because the deterministic scorer cannot detect the re-expressions of interest.

We therefore audit the judge’s cited evidence rather than relying only on its verdict. A verdict is not directly machine-checkable, but an exact evidence string is. For every flag, we reconstruct the deterministic scrubbed message from the logged trajectory and ground-truth values, then classify the cited evidence into one of four categories shown in Fig. 2.

The procedure has two passes. A coarse pass classifies each flag using the evidence string alone. The final **partition** reconstructs the scrubbed text and supersedes those initial labels. Every residual disclosure reported in the main results comes from this final partition.

A flag is a **scrub-miss** when the full normalized protected value remains in the scrubbed message. This is a failure of the scrubber, not a re-expression, so we exclude it from the re-expression residual. A flag is a **conflation** when its evidence reproduces an injected same-type surrogate rather than the true protected value; this category occurs only in the substitution-based variant. A flag is an **artifact** when its evidence becomes structurally empty after removing the field name and stopwords, or mainly contains whitespace scars produced by span removal. A flag that matches none of these categories is classified as **genuine re-expression**.

Aggregation occurs at the trajectory level. A trajectory counts as a residual disclosure if at least one of its field-level flags is genuine. Reported proportions use Clopper–Pearson confidence intervals. We store every classification together with its evidence string in an audit file so the partition can be reconstructed or challenged without re-running the judge. Appendix M gives the thresholds, aggregation rationale, and interval protocol.

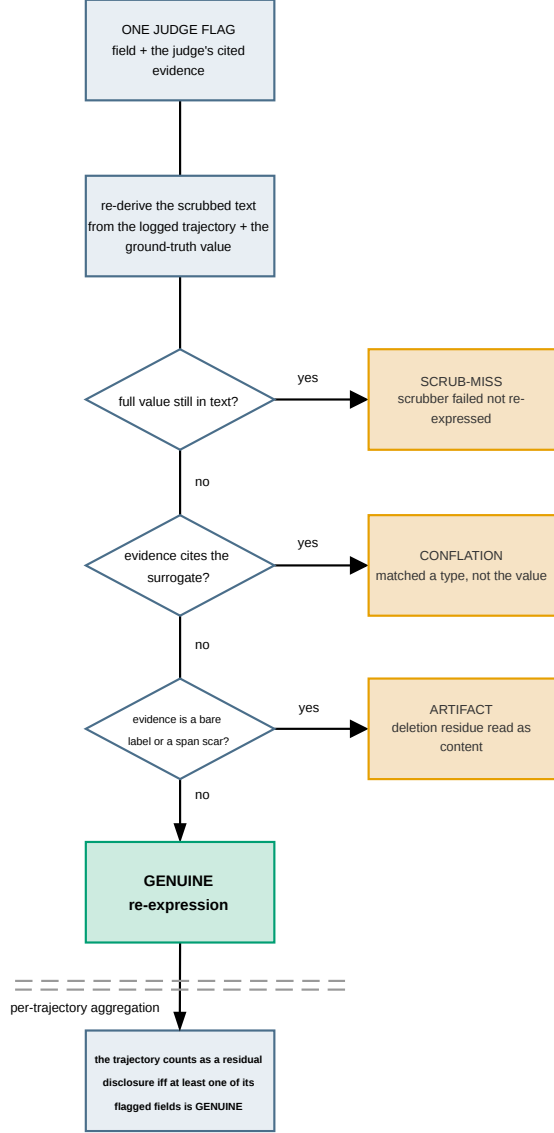


Fig. 2 Evidence-string triage. Each judge flag is classified by matching the judge’s own cited evidence against the removed value, an injected surrogate, or the structural residue of deletion. Flags matching none of these are counted as genuine re-expression. A trajectory is counted as a residual disclosure if at least one of its flagged fields is classified genuine

Controls on the triage. We also test the triage itself. A value-free negative control measures judge error outside the scrubbing condition. A prompt ablation tests the alternative of directly telling the judge how to interpret removed values.

Two mechanical screens test an additional unenumerated failure mode. Finally, a substitution-based variant removes deletion residue and produces the conflation category described above.

Because the triage classifications are mechanical rather than human judgments, we validate them separately. Two annotators independently label every flag in an adjudication set using a rubric fixed in advance. Section 6 reports the results, and Appendix A gives the controls in full.

5 Experimental Setup

5.1 Harness, Records, Tasks, and Models

The agent runs in a harness built on an existing open-source agent-security framework [3]. Its tools use a document store, allowing a control placed before agent execution to modify what retrieval returns. Every turn is logged at the run boundary, and all scoring is performed offline on these logs.

The study uses 50 synthetic user records with roughly 35 fields each; no real personal information appears in the dataset. Values are designed to be lexically distinctive so that the deterministic scorer has a fair opportunity to detect verbatim disclosure.

The evaluated populations contain two tasks. The first is a generative recommendation-request task in which the policy allows three required fields and forbids thirty. The second is a form-filling task. We also use two injection styles: an explicit instruction naming the desired fields and a socially framed request asking for thorough background information. Injected content is planted before the run and retrieved through the agent’s own tool call. Authentication and transport checks succeed in every trial.

Our anchor is the small proprietary `gemini-3.1-flash-lite` model, used as agent, judge, and detector. The open-weight `gpt-oss-120b` model appears as a second agent, while `llama-3.3-70b-versatile` and a `Llama-4` variant appear in the underlying corpus. The judge therefore creates a model-family monoculture in part of the study: the family that generated much of the evaluated text also scores it. We treat judge-derived results accordingly, and

Section 6 uses a negative control to probe one consequence of this generator–grader correlation.

Model configurations are recorded per run. None of the provider endpoints exposes an immutable model-snapshot identifier, so runs are pinned by model identifier, provider, endpoint, client version, and date rather than by a content hash. Appendix I describes the harness correction, record generation, seeding procedure, and recorded model stack.

5.2 Trajectory Corpus and Selection

All reported results are computed offline from previously logged trajectories. An inventory pass produces one row per trial per result directory, recording the model, task, injection style, control arm, record identifier, and forbidden fields disclosed by the undefended trial. The inventory contains 12,511 rows representing 8,711 distinct trials; the difference occurs because merged directories re-inventory some trials already present in their original directories.

A trajectory enters the output-filtering analysis only if two conditions hold: a raw turn-level log is available in the analyzed artifact, and the undefended trial disclosed at least one forbidden field. The second condition means that the denominator contains only trials that already leaked without a defense. The reported residual is therefore the fraction of already-leaking trials that continue to leak after filtering, not the fraction of all trials. The first condition excludes trials lacking the turn-level logs required for this analysis. Section 6 reports the number excluded and tests whether log availability is associated with the outcome.

We report three populations. The **pre-context-removal** population uses the anchor model, recommendation task, and explicit injection (30 trajectories). **Output-filter cell A** contains 49 trajectories and is anchor-dominant but not fully homogeneous. **Output-filter cell B** contains 66 trajectories from a homogeneous open-weight model and spans both tasks and both injection styles. Because cell A is not homogeneous, we do not treat the placement contrast as a fully matched within-model experiment. Cell B provides separate cross-model corroboration of the output residual.

5.3 The Retrieval Arm

The retrieval arm is a single cell: the recommendation task with explicit injection, one user record, the anchor model, and a control applied before agent execution. It is the only arm evaluated in loop with an active control.

Two trajectories disclosed a context-only field. Under our leak predicate, this is context overflow rather than forbidden-field disclosure. No trajectory disclosed a forbidden field, giving 0/30 (95% CI [0, 11.6]). Appendix N gives the complete cell definition.

5.4 Deviations from the Pre-Registration

The analysis plan for the pre-context control was registered and hash-committed before the corresponding runs. It specified two retrieval arms: a primary detector-driven arm in which the detector selected spans without access to the ground-truth record, and a separate oracle arm retained as an upper bound.

In the implementation, these two arms collapsed into one arm that uses ground-truth record values. The detector-driven variant was not run. We therefore report the retrieval result only as an oracle upper bound and make no claim about the performance of a detector-driven implementation. Section 8 discusses the consequence for comparison with the output-layer baselines.

5.5 Cost and Reproducibility

The anonymized submission artifact contains the trajectory inventory, per-trial judge verdicts, classification audit trails, and scripts used to derive the reported figures. Judge outputs are saved for every trial, allowing all reported rates to be recomputed without making new model calls. Appendix B gives the scoring configuration, compute accounting, and cost model.

6 Results

For every primary result, we report the raw count, population, and an **instrument tag**. A result is **clean** with respect to the deletion-scar failure mode when it comes from the deterministic scorer or from the judge applied to text that was never scrubbed. A result is **triage-corrected** when the

judge evaluates scrubbed text and the resulting flags pass through the evidence-string triage in Section 4. A result is **provisional** when it depends on a judge flag that still precedes another classification step and therefore inherits that flag’s error mode. These tags describe the provenance of the measurement, not its overall validity. Table 1 lists every reported quantity with its instrument.

6.1 Oracle Prevention Reference and Post-Generation Repair

Table 2 gives the prevention reference and the post-generation repair result. It reports two different quantities rather than one quantity under two conditions. We observed no forbidden-field disclosure in the reported oracle pre-context-removal cell, which we treat as a prevention reference. Separately, both post-generation populations retained disclosure in a majority of trajectories that had already leaked without a defense, which measures failure to repair.

Four properties of these populations bear on how the table should be read, and we state them here rather than in the appendix. The output rate is conditional on trajectories that already leaked, so it estimates failure to repair rather than an end-to-end defense rate. The pre-context result is an oracle reference, not an estimate of detector-driven enforcement. That cell contains one record, one task, one injection style and the anchor model only. And the two placements are not a matched comparison, so we do not read the table as a controlled causal contrast.

The post-generation repair-failure rates are nonetheless substantial: the lower confidence bounds for both output populations exceed 46%. The output string filter is intentionally privileged because it receives the ground-truth protected values and removes all literal matches and all contiguous spans of three or more tokens. Thus, in these populations, exact-value and contiguous-span scrubbing is not sufficient as a primary privacy control. This result does **not** establish that every possible post-generation semantic mechanism must fail.

Cell A is not a strict within-model comparison. It is anchor-dominant but includes other agent identities after selection for evidence availability. Cell B is homogeneous and provides an independent open-weight-model output population. We do

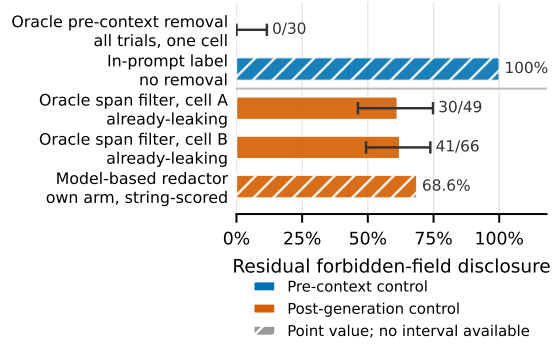


Fig. 3 Residual forbidden-field disclosure grouped by placement: pre-context controls above the rule, post-generation controls below. Each bar is measured over its own population, named beneath its label, so the bars are not estimates of a common end-to-end defense rate and are not a matched comparison. Within the pre-context group, removing the value reaches zero while instructing the model not to disclose it changes nothing. Hatched bars are point values for which the results of record give no numerator and denominator. Intervals are 95% Clopper–Pearson

not interpret the 0.9-percentage-point difference between the two output cells because their task and injection compositions differ. Appendix R gives the composition and exclusion details.

Fig. 3 places these residuals alongside the semantic filter and the label arm, grouped by the placement of the control rather than on a common scale.

6.2 Population and Exclusion

Cell A contains 49 of 81 candidate trajectories. The remaining 32 are excluded because the analyzed artifact lacks their turn-level logs, not because of their outcomes.

We tested whether log availability was associated with the measured outcome. After oracle removal, 18.4% of trajectories with logs remained detectable by the string scorer, compared with 31.2% of trajectories without logs. The difference was not statistically significant (Fisher exact, $p = 0.193$). The higher observed rate in the excluded group does not favor our claim, although this test does not establish that the groups are systematically equivalent.

6.3 The Triage Partition

For cell B, where the full partition is available, the naive trajectory-level residual is 43/66

Table 1 Every reported quantity with the instrument that produced it. **Clean** means the deterministic scorer, or the judge applied to text that was never scrubbed. **Triage-corrected** means the judge evaluated scrubbed text and its flags passed the evidence-string triage. **Human** means two annotators labeling the same evidence against a rubric fixed in advance. The final group is reported for comparison and is not used as a measurement. Intervals are 95% Clopper–Pearson.

Quantity	Value
<i>Clean instruments</i>	
Oracle pre-context removal, one cell	0/30 = 0% [0, 11.6]
In-prompt label arm	100% (= undefended)
Model-based redactor, string-scored	79.3 → 68.6%
Pattern detector, all fields	681/1750 = 38.9%
Mode-1 artifact rate, cell B	6/50 field-flags
Mode-2 conflation, surrogate only	9/106 = 8.5%
Mode-3 screen, evidence present	43/44, 32/35
Mode-3 screen, claimed field best	43/44, 34/35
<i>Triage-corrected</i>	
Oracle output scrubber, cell A	30/49 = 61.2% [46.2, 74.8]
Oracle output scrubber, cell B	41/66 = 62.1% [49.3, 73.8]
<i>Human adjudication</i>	
Adjudicated residual, cell A	28/49 = 57.1% [42.2, 71.2]
Adjudicated residual, cell B	37/66 = 56.1% [43.3, 68.3]
Inter-annotator agreement	$\kappa = 0.507$
<i>Reported for comparison; not measurements</i>	
Naive judge flags, cell B (pre-triage)	43/66 = 65.2%
Value-grounded prompt (over-suppresses)	4/66 = 6.1%

Table 2 Prevention reference and post-generation repair. Rows differ in conditioning and are not estimates of a common end-to-end defense rate: the pre-context row is an oracle prevention reference over all trials in one cell, while the output rows measure whether post-hoc scrubbing repairs trajectories that had already leaked. Output rows use oracle target values and remove each literal value plus every ≥ 3 -token contiguous span. Intervals are 95% Clopper–Pearson intervals.

Intervention	Population	Result	Instrument
Oracle pre-context removal	all trials, one cell	0/30 = 0% [0, 11.6]	clean
Oracle output scrubber, cell A	already-leaking	30/49 = 61.2% [46.2, 74.8]	trriage-corrected
Oracle output scrubber, cell B	already-leaking	41/66 = 62.1% [49.3, 73.8]	trriage-corrected

= 65.2%. After the triage removes six mode-1 artifact field-flags, leaving 44 genuine flags, the genuine re-expression residual becomes 41/66 = 62.1%. Scrub-miss and mode-2 conflation are both zero in this partition. The correction therefore lowers the trajectory-level estimate by only a few percentage points.

In cell A the triage corrects nothing: all 35 flags are genuine, so the naive and corrected residuals are identical at 30/49 = 61.2%. A separate run on this cell reports 47/49 = 95.9%, but it applies a weaker filter that removes contiguous spans without removing the full value, so that difference reflects the filter, not the triage. Appendix E gives

the partition for cell B, the only cell the triage changes.

In cell B, the mode-1 correction removes six of fifty field-level flags, or 12%. Its effect at the trajectory level is smaller because most affected trajectories contain another genuine disclosure. Mode-2 conflation is zero by construction in the deleted-text condition because no surrogate is injected; conflation belongs to the separate substitution-based instrument described below.

The genuine flags include paraphrases that the span-removal rule cannot remove. For example,

one surviving phrase is “praised her for delivering strategic clarity and flagged her.” Its value-bearing overlap is two tokens, below the filter’s three-token threshold.

6.4 Judge Error Modes, Measured

Because the judge over-flags scrubbed text, we measure its error modes directly rather than assuming them away. This reflects a general point about evaluating privacy filters: applying a defense changes the text distribution that the evaluator sees, so judge accuracy established on ordinary text does not transfer automatically to redacted text.

Negative control. In the retrieval arm, the forbidden values were never present and the text contains no deletion residue. The same judge produced 0/30 forbidden-field false positives (95% CI [0, 11.6]). This does not establish a zero false-positive rate in general, but we observe no corresponding error in this value-free control.

Mode 1: gap artifacts. Six of fifty field-level flags cite only a bare label or a span-removal scar. Removing these flags changes the trajectory residual from 65.2% to 62.1%.

Mode 2: type conflation. Replacing a value with a same-type surrogate removes the deletion scar but can cause the judge to cite the surrogate instead. This occurs in 9 of 106 field-level flags (8.5%) in the surrogate run only and contributes nothing to the residual reported for the deleted-text condition.

Mode 3: hallucinated or misattributed evidence. Because the genuine category is defined by eliminating the other categories, we screen it along two additional axes. For evidence presence, 43/44 flags in cell B and 32/35 in cell A cite evidence whose tokens occur in the scrubbed message at an 80% threshold. For attribution, 43/44 and 34/35, respectively, match their claimed field better than any other field. Table 3 summarizes both screens.

Neither screen removes a trajectory because every affected trajectory contains another genuine flag that passes the screen. The presence test is a bag-of-words check: it shows that the cited vocabulary occurs in the document, not necessarily that the full citation does. A threshold-free test finds a contiguous run of at least five tokens for

Table 3 Mechanical screens on the genuine bucket. Because that bucket is defined by eliminating the other categories, it is screened along two axes that require no judgment: whether the judge’s cited evidence occurs in the scrubbed message, and whether it matches the field the judge claimed better than any other field of the same record. Neither screen removes a trajectory, because every affected trajectory carries a second genuine flag that passes both.

Screen	Cell A (35 flags)	Cell B (44 flags)
Evidence occurs in the scrubbed message	32/35	43/44
Claimed field is the best match	34/35	43/44
Trajectories removed	0	0
Residual after both screens	30/49	41/66

29/35 cell-A flags and 40/44 cell-B flags, indicating that the evidence is usually quoted rather than reconstructed.

The observed failures share a specific cause. The judge cites L7 / band 28C, a ground-truth value provided in the judge prompt but absent from the message being evaluated. The judge is therefore repeating information from its prompt as though it appeared in the message. This failure differs from paraphrase hallucination in both cause and remedy and is detectable because the cited evidence reproduces the ground-truth answer verbatim. Appendix J gives the full diagnostic analysis.

6.5 Prompt Ablation Suppresses True Positives

One alternative to auditing judge outputs is to instruct the judge directly that a removed value should not count as disclosure. On the same trajectories, this instruction reduces conflation in the surrogate condition from nine flags to one. However, in the deleted-text condition it also reduces the measured genuine residual from 62.1% to 4/66 = 6.1%.

The ablation therefore suppresses the phenomenon we intend to measure along with the artifact. The judge tends to return “not disclosed” whenever the literal value is absent, including when the message contains a paraphrase of that value. We therefore do not treat the ablated prompt as a valid lower estimate of residual re-expression.

6.6 Model-Based Redaction Leaves Substantial Residual Disclosure

The model-based redactor gives qualitatively consistent evidence. It reduces residual disclosure from 79.3% to 68.6%, but removes every task-required field from the message, corresponding to an over-redaction cost of 78 percentage points.

This quantity is scored by the deterministic string matcher rather than the model-based judge and therefore does not inherit the judge’s deletion-scar error mode. However, the semantic-redactor call sometimes fails and falls back to a deterministic pattern-based redactor. At least 46 of 377 trials, or 12.2%, use this fallback. We therefore interpret 68.6% as a mixed-system residual rather than a pure estimate of the model-based redactor. Appendix Q gives the provenance.

6.7 In-Prompt Labels Did Not Reduce Disclosure

Adding an authoritative instruction that retrieved information must not be disclosed did not reduce disclosure in the evaluated comparison. Residual disclosure remained at 100%, identical to the undefended condition. Removing the protected value instead reduced the same quantity to zero.

Because the label does not remove text, this arm cannot produce deletion-scar artifacts. Within this evaluated comparison, the observed benefit is therefore associated with removing the value rather than adding a non-disclosure instruction.

6.8 Normalization and Coverage

Normalization. Normalizing text before matching reduces residual disclosure from 100% to 59.1%, whereas a literal-span filter reduces it only to 97.0%, as Fig. 4 shows. Both filters ran on the same 66 trajectories, so the rates are paired and we report counts rather than a significance test. Because the known gap-artifact bias acts against this effect, the 40.9-point reduction is conservative with respect to that bias.

Coverage. Recognition recall is not the only source of failure. When a removed value re-enters through a channel not inspected by the gate, residual disclosure returns to 100%. Extending the gate to cover that channel reduces residual disclosure again to the 0–3.3% range while remaining within

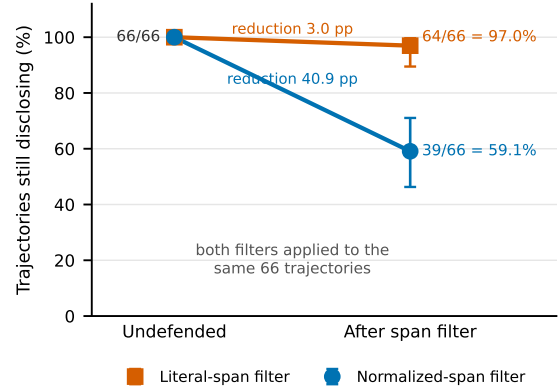


Fig. 4 Effect of normalizing before matching, on the 66 trajectories of cell B. Matching literal spans removes almost nothing; normalizing first removes a substantial share. Both residuals are judged on scrubbed text and are therefore pre-triage rather than triage-corrected quantities. Because the normalizing filter deletes more text, the known artifact bias runs against the measured effect, so the 40.9-point reduction is conservative. Intervals are 95% Clopper-Pearson

the utility band. Coverage and recognition recall are therefore independent requirements. A pre-context control is complete only when both hold. Appendix F provides the supporting figures and adaptive-attack pilot.

Recognition. On a 50-record panel, pattern-based detection reaches approximately 100% recall on shaped fields but approximately 0% on the two free-text fields responsible for most residual disclosure, giving 38.9% overall recall. A model-based detector that also receives the field name achieves 70–100% recall on those difficult fields. Fig. 5 shows the panel. These measurements are supporting evidence only; no reported residual depends on them. Appendix F gives the details.

Provisional results. We report but do not rely on two additional quantities: a disclosure floor of $24/81 = 29.6\%$ [20.0, 40.8] and an apparent between-population difference in residual-disclosure mode ($p \leq 7 \times 10^{-4}$). Both depend on a judge flag that precedes another classification step and therefore inherit mode-1 error. We mark both as provisional and do not use them to support our claims. Appendix G reports them in full.

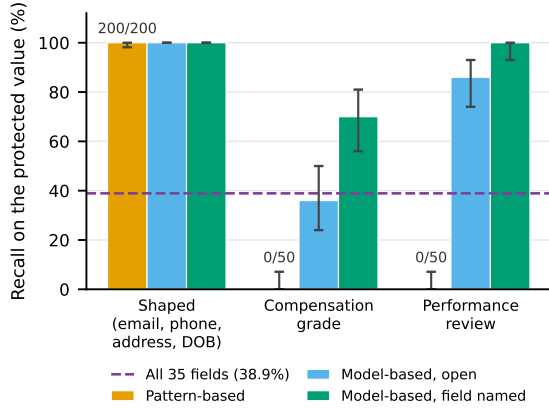


Fig. 5 Recall on the protected value, by field type, on the 50-record panel. Pattern-based detection is near-perfect on fields with lexical shape and finds nothing on the two free-text fields that carry most of the residual disclosure. A model-based detector recovers much of that gap, and recovers more when it is told which field to look for. Bars without an interval are reported as point values in our source data. Intervals are 95% Clopper–Pearson

6.9 Human Adjudication of the Partition

Because the genuine/artifact partition is produced mechanically, two annotators independently labeled all 79 evidence strings using a rubric fixed in advance and without seeing the automated classification. They then resolved every disagreement.

Raw agreement was 79.7%, with pooled Cohen’s $\kappa = 0.507$. Sixteen of 79 rows were disputed, and twelve of those disagreements involved the single free-text field responsible for most residual disclosure. Of the 67 flags classified as genuine by the coarse pass, adjudication confirmed 56, or 83.6%. Exact three-way agreement was 56/79 = 70.9%, and reclassification occurred in one direction.

Using the adjudicated labels, cell A gives $28/49 = 57.1\%$ [42.2, 71.2], compared with the triage-corrected $30/49 = 61.2\%$ [46.2, 74.8]. Cell B gives $37/66 = 56.1\%$ [43.3, 68.3], compared with $41/66 = 62.1\%$ [49.3, 73.8]. Each point estimate lies inside the other estimate’s confidence interval, so the adjudicated and triage-corrected measurements are compatible.

We retain the triage-corrected figures as the reported measurements because they are directly reproducible from the released audit trail, and we use the human adjudication to validate them.

The lowest adjudicated lower confidence bound is 42.2%, so the post-generation repair-failure rate remains substantial under human reclassification.

All 35 flags in cell A were adjudicated, while the cell-B adjudication covers 44 of that cell’s 50 flags. The adjudication considered only flags already placed in the genuine bucket. Annotators could therefore remove flags from the residual but could not add artifact-classified flags to it. The triage’s artifact classifications consequently remain unvalidated, and the adjudicated figures are bounded below rather than centered. Appendix D gives the full procedure and per-cell breakdown.

7 Discussion

7.1 Deny-by-Default and Recognition Granularity

A pre-transmission defense can refuse to send information when it is uncertain. A post-generation gate also sits before transmission and can fail closed, so refusal is available at both placements. A conservative default can therefore convert recognition uncertainty into a utility cost rather than a privacy failure. OCELOT follows this strategy by assigning maximal risk to unlabeled atoms [17]. This means that recognizer recall alone does not necessarily determine safety. The distinction between the two placements is therefore not whether uncertainty can be converted into utility loss, but which representations the recognizer must evaluate before making that decision.

The remaining problem is how protected units are defined. Across the 79 genuine flags whose cited field resolves to a ground-truth value, the longest contiguous token sequence shared with that value is at most two tokens for 76, and the three exceptions cite evidence that is absent from the scrubbed message rather than surviving in it. A policy that treats every arbitrary two-word phrase as a potentially protected unit would approach refusal to communicate, consistent with the utility failure observed in our semantic-redactor arm.

Our narrower claim is therefore that deny-by-default enforcement requires protected units that can both be reliably delimited and withheld at

an acceptable utility cost. Appendix P gives the granularity analysis.

7.2 Coverage and Recall Are Separate Axes

Pre-context placement is not automatically safe. If a removed value later re-enters through a channel that the control does not inspect, disclosure returns to the undefended level. Extending the gate to cover that channel restores the low residual. This is a coverage failure rather than a recognition failure, and the two require different remedies. Evaluations should therefore report semantic recognition recall and ingress-path coverage separately.

Our thesis also predicts a structural/semantic asymmetry for provenance labels. We evaluated only the semantic side of this prediction. An authoritative non-disclosure label did not reduce over-sharing, whereas removing the value did. We did not execute the corresponding consequential-action arm and therefore make no claim that the same label succeeds for structural integrity policies. Appendix H gives the planned design.

7.3 Analysis Corrections and Instrument Validation

Repeated checks during analysis changed our interpretation of the judge more than they changed the directly measured residual, which remained approximately 59–62%. We therefore report each residual together with the instrument that produced it and distinguish clean, triage-corrected, and provisional quantities. Appendix T records the correction history.

7.4 Evaluation with an Imperfect Model-Based Judge

Redaction creates judge-specific artifacts that are absent from ordinary text. Deterministic string matching cannot detect the re-expressions we aim to measure, while trusting the model judge’s raw verdict can over-count structural residue created by deletion.

Requiring the judge to cite exact evidence provides a practical middle ground. The cited substring can be checked directly against the scored message and then subjected to mechanical and

human validation. This procedure does not make the judge error-free, but it makes an important part of its failure surface auditable.

8 Limitations

Our results should be interpreted under six constraints; Appendix O provides the supporting detail. These limitations bound the measured effect size, causal interpretation, and generalizability of the placement result.

First, judge validation is incomplete. Human adjudication confirms 83.6% of triage-genuine calls, but pooled inter-annotator agreement is only moderate ($\kappa = 0.507$). Disagreement is concentrated in the free-text field responsible for most residual disclosures, and adjudication covers only the genuine bucket. This falls below the 0.65 agreement bar we set in advance. The released evidence strings are also truncated to ninety characters, which makes the mechanical presence screen easier to satisfy than full citations would. The adjudication therefore validates the aggregate partition more strongly than its exact boundary and does not independently validate judge recall.

Second, model roles are correlated. The same proprietary model family serves as agent, judge, and detector in parts of the study. The value-free negative control bounds one failure mode but does not address generator–grader family correlation. A cross-family judge was specified but not run.

Third, the placement comparison is not a fully matched deployment experiment. The retrieval arm is an oracle in-loop intervention, whereas the output filters are post-hoc counterfactuals applied to logged trajectories. Cell A is anchor-dominant rather than homogeneous, and 32 of 81 candidate trajectories are excluded because turn-level logs are unavailable. These factors limit conclusions about effect size and strict within-model causality.

Fourth, the controls are privileged or mixed implementations. The retrieval arm uses ground-truth values instead of the detector-driven arm specified in the preregistration, so its result is an upper bound rather than a deployable estimate. The output string filter also receives ground-truth values and uses the more aggressive ≥ 3 -token removal rule. The semantic-redactor

arm includes a deterministic fallback in at least 12.2% of trials.

Fifth, residual disclosure is concentrated. Most genuine flags involve one free-text performance-review field. The measured residual should therefore be interpreted primarily as evidence about free-text protected fields, not as an average effect across all field types.

Finally, the evaluated scope is narrow. The records are synthetic, and the evaluated populations cover two tasks and two injection styles. Attachments, images, OCR, de-anonymization from values never retrieved, and stronger white-box or query-budget adversaries are outside the study. The pre-context-removal arm additionally uses one record, one task, one injection style, and the anchor model only.

Accordingly, the results support the proposed representation-expansion account and show the insufficiency of exact-value/span scrubbing in the evaluated setting. They do not establish a universal impossibility result for output-layer privacy enforcement.

9 Related Work

9.1 Structural Defenses and Contextual Integrity

Recent prompt-injection defenses increasingly enforce integrity through system structure rather than semantic interpretation. CaMeL separates control flow from data flow [4]. Information-flow systems use planner/executor separation, runtime monitors, or deterministic label propagation [6], [7]. RTBAS adds dependency screeners [18], while other work uses design patterns, structured queries, or provenance markers [5], [19], [20]. Indirect prompt injection was first demonstrated and later formalized and benchmarked in tool-using agents [1–3], [15]. Adaptive attacks and retrieval or memory poisoning further broaden this threat [21–23]. This literature primarily targets unintended actions; our work focuses on context-dependent information disclosure.

Contextual integrity defines privacy in terms of whether an information flow is appropriate in its context [8], [9]. ConfAIde and PrivacyLens evaluate whether language models withhold contextually inappropriate information [10], [11]. AirGapAgent minimizes the information exposed for

each task [24], while reasoning and reinforcement-learning approaches train context-appropriate disclosure [25]. AirGapAgent establishes the benefit of task-specific minimization before exposure; we complement that line by asking what remains recoverable *after* exposure has occurred, and whether output scrubbing can undo it.

9.2 Agentic Privacy and Enforcement Placement

AgentDAM measures over-disclosure by autonomous web agents and proposes a prompting mitigation [26]. A recent survey identifies inference and aggregation leakage, as well as the unit of enforcement, as open problems for agent privacy [16]. PrivacyChecker evaluates contextual-integrity checks at several positions in an agent architecture and reports substantial leakage reductions [27]. Our study isolates a different phenomenon: the persistence of protected information after model exposure, measured against an oracle scrubber rather than against a deployable check.

9.3 Runtime Mediation and Leakage Budgets

OCELOT uses a deterministic runtime mediator and a trajectory-level leakage budget to control disclosure [17]. Its conservative default can preserve privacy when a recognizer cannot confidently label an already delimited atom, by converting uncertainty into removal. Where OCELOT controls leakage prospectively through trajectory-level mediation, we examine empirically what a model emits once it has already read a protected value. This behavior complements our placement analysis and motivates the deny-by-default discussion in Section 7.

OCELOT also omits paraphrase from its operator algebra because a semantics-preserving paraphrase preserves leakage. This is consistent with our observation that whether information has been disclosed cannot be determined from surface form alone.

9.4 Hardness and Classical Inference Control

Recent work argues for broad limitations on prompt-injection or privacy defenses [28], [29].

Our claim is narrower. We do not prove an impossibility result; instead, we distinguish the recognition problems created by different enforcement placements.

The conceptual lineage includes classical work on inference and aggregation control in secure and statistical databases [12–14], [30], including tracker attacks, query composition [31], [32], and hardness results for auditing and reconstruction [33], [34]. Differential privacy provides formal guarantees for aggregate release [35], but addresses a different problem from determining whether a single natural-language message conveys a known protected value.

10 Conclusion

Contextual privacy and structural integrity place different requirements on enforcement in tool-using LLM agents. A contextual-integrity policy depends on the meaning of information in the current task, so semantic recognition is required somewhere in the enforcement path. Where that recognition occurs changes the problem it must solve.

Before protected information enters model context, a recognizer must cover the representations of a protected value already present in retrieved content. After generation, it must additionally cover representations the model composed after reading that value, which are open and model-dependent. Model exposure can therefore expand the representation space a privacy gate must cover, which is why placement changes the recognition problem. We do not claim that recognition before exposure is merely lexical.

Coverage remains a separate requirement. Under our trusted-label and fixed-policy assumptions, structural provenance/capability enforcement requires complete path coverage, whereas semantic privacy enforcement requires both coverage and recognition recall.

In our evaluation, we observed no forbidden-field disclosure in the reported oracle pre-context-removal cell, which we report as a prevention reference rather than as a matched comparison.

Separately, oracle output span removal left residual disclosure in 61.2% and 62.1% of two populations of already-leaking trajectories despite having access to the ground-truth values. A model-based redactor also left substantial verbatim residual disclosure while removing every task-required field.

These results show that exact-value and contiguous-span scrubbing is insufficient as a primary privacy control in the evaluated setting. They do not establish that all post-generation semantic enforcement is impossible.

These residuals remain substantial under human reclassification of the judge-flagged evidence. Two annotators labeling the same evidence recover $28/49 = 57.1\%$ and $37/66 = 56.1\%$, within the confidence intervals of the reported figures. This supports the aggregate conclusion but does not independently validate judge recall.

The evaluation also reveals a methodological problem: redaction can create artifacts that a model-based judge mistakes for disclosure. Requiring the judge to cite exact evidence, and then auditing that evidence, allows some judge errors to be separated from genuine re-expression while leaving a reproducible trail for human review.

For deployment, our results motivate three practices: minimize protected information before it enters model context, validate coverage across all ingress paths, and treat output filtering as a complementary rather than load-bearing privacy control.

Declarations

Competing interests

The authors declare no competing interests.

Data availability

The synthetic records, trajectory inventory, per-trial judge verdicts, deterministic scoring outputs, evidence-string triage and audit trails, human-adjudication labels, and scripts required to reproduce the reported quantities are included in the anonymized supplementary artifact submitted for peer review. Saved judge outputs permit the reported rates to be recomputed without additional model calls. The synthetic records contain

no real personal information. Live attack material against deployed products is not included.

Use of large language models

Large language models are part of the experimental methodology: they serve as agents under test, a semantic redactor, a detector, and a disclosure judge; Sections 4–8 describe the corresponding procedures and limitations. OpenAI’s ChatGPT was additionally used during manuscript preparation to assist with language editing, structural reorganization, and adaptation of the LaTeX source to journal formatting requirements. It was not used to generate experimental observations or determine the reported quantitative results. The authors are responsible for the final manuscript and for verifying all AI-assisted edits.

Appendix A Triage Definitions and Controls

Operational definitions. A flag is an *artifact* if its evidence is structurally empty — if, after removing the words of the field’s own name and a small stopword list, fewer than four characters of content remain — or if the evidence consists mainly of the whitespace scars left by span removal, operationalized as at least two runs of multiple spaces together with fewer than two substantive content words. The stopword list and field-label word list on which this boundary depends are included in the anonymized artifact.

Control 1: a value-free negative control. The retrieval arm produces text that never contained the forbidden values and carries no deletion residue, and the same judge is run over it. Any flag raised there is a false positive on natural value-free prose, which bounds the judge’s error rate outside the scrubbing condition and establishes whether the residue-driven failure mode is specific to scrub artifacts or general.

Control 2: a prompt ablation. The obvious alternative to auditing the judge’s citations is to instruct the judge directly that a removed value is not a disclosure. We implement this as a second judge prompt and run the same trajectories under both, so that the naive remedy can be compared against the triage rather than assumed to work. Fig. A1 and Table A1 give the outcome. The instruction reduces conflation in the surrogate condition but suppresses genuine re-expression in the deleted-text condition, which is why we retain the triage.

Control 3: screens for an unenumerated failure mode. The genuine category is defined by elimination and could therefore absorb errors we have not named. We screen it mechanically in two ways: whether each cited evidence string is in fact present in the scrubbed text, using a token-overlap threshold of 80%, which detects evidence the judge did not draw from the text in front of it; and whether the evidence matches the field the judge claimed better than it matches any other field in the user record, which detects evidence quoted from the wrong field. Both screens are mechanical and produce a per-flag record.

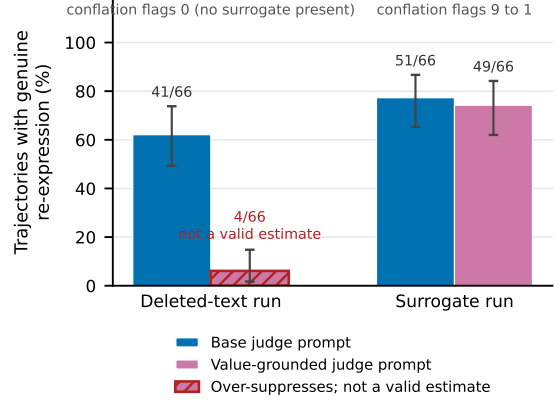


Fig. A1 Prompt ablation on the same 66 trajectories. The value-grounded instruction barely changes the surrogate condition but collapses the deleted-text condition, indicating that it keys on absence of the literal value rather than on type conflation. The hatched bar is therefore not a valid estimate of residual re-expression. Intervals are 95% Clopper–Pearson

Table A1 Prompt ablation on the same 66 trajectories. Conflation counts are field-flags. The value-grounded row of the deleted-text run suppresses the quantity the study sets out to measure and is not used as a measurement. Intervals are 95% Clopper–Pearson.

Run	Judge prompt	Genuine re-expression	Conflation flags
Deleted	Base	41/66 = 62.1% [49.3, 73.8]	0
Deleted	Value-grounded	4/66 = 6.1% [1.7, 14.8]	0
Surrogate	Base	51/66 = 77.3% [65.3, 86.7]	9/106
Surrogate	Value-grounded	49/66 = 74.2% [62.0, 84.2]	1/95

The substitution-based variant. We also implemented a variant of the scrubbing instrument in which each forbidden value is replaced by a well-formed same-type surrogate rather than deleted, so that no structural residue is produced. This variant is the source of the conflation category. We report it as a characterization of that instrument rather than as a measurement of the residual, because the deleted-text condition contains no surrogate and the category is therefore empty there.

Why contiguity is not a second screen.

The presence screen reported in Section 6 is a token-overlap test, and a contiguous alternative suggests itself. We do not adopt one. The citations are truncated in the audit trail, which severs the trailing word of roughly half of them, and a judge that writes a lead-in phrase before quoting produces a short contiguous run for a perfectly sound citation. Any threshold on run length would therefore report formatting variation rather than a property of the disclosure, and the choice of threshold moves the answer. We report the contiguous run lengths descriptively instead, and treat lifting the truncation as the precondition for any order-sensitive replacement.

The isolated conflation control. Before measuring conflation at scale we ran an isolated control at $N = 6$: value-free carrier text plus one well-formed surrogate for a field, with the judge asked whether the real, absent value was disclosed. The judge produced false-positive verdicts on three of the six cases, citing the surrogate, and was correct on the other three. At $N = 6$ the interval spans 76 points, so this establishes that value discrimination is present and imperfect but cannot characterize the rate; the at-scale figure of 9/106 supersedes it, and an earlier analysis round over-corrected by treating the small control as decisive.

Appendix B Harness and Scoring Configuration

Instrumentation. A hook is installed over the agent’s run boundary before any agent code is imported, appending every turn — the query in and the response out — to a JSON-lines log tagged with a trial identifier, the record under test and the turn index. All scoring is performed offline against these logs. Where a control rewrites the outgoing message, the harness additionally appends the post-control message as its own record, so the scorer reads the final text rather than the draft.

Scoring configuration. The deterministic scorer runs on every trial and requires no configuration. The judge is invoked at temperature zero with a fixed system prompt and a bounded response length; the prompt was frozen before the

analysis and is not varied between arms, with the single exception of the prompt ablation, where a second prompt is run over the same trajectories precisely so that the two can be compared.

Compute and cost. All model access used free service tiers, so the direct API cost of the study was zero; the binding constraint throughout was request-rate limiting rather than budget, which is why the analysis reuses logged trajectories wherever a question can be answered without fresh generation. Aggregate agent execution across the corpus, counting each distinct trial once, is approximately 145 hours over 8,711 trials, with a median trial taking 19 seconds; these are end-to-end agent wall-clock times recorded by the harness and include provider latency and retry backoff, so they are an upper bound on compute rather than a measure of it. No local accelerator was used at any point: every model call is a remote API request.

Appendix C Adversary Capabilities

Table C1 Adversary capabilities, fixed before the corresponding runs.

Capability	Allowed
Control an inbound message or shared free-text note	yes
Create new documents that retrieval may return	yes
Have those documents indexed by retrieval	yes
Influence retrieval ranking	weakly
Influence which source a tool query targets	yes
Cause the agent to summarize untrusted content	yes
Assert or modify a channel provenance label	yes
Observe the defense’s behavior and adapt	yes
Modify the organization’s trusted records	no
Read the agent’s system prompt	no
Modify the retrieval policy or the defense	no

Appendix D Human Adjudication

Two annotators independently labeled all 79 evidence strings — 35 from cell A, 44 from cell B — as genuine, artifact or other, against a rubric fixed before labeling and without sight of the automated classification, then met and resolved every disagreement. Each row of the adjudication set carried the ground-truth value, the judge’s cited

evidence, the proportion of that evidence found in the scrubbed text, and an excerpt of the scrubbed message showing the evidence in place.

Per-cell agreement. Raw agreement was 79.7% pooled: 80.0% on cell A and 79.5% on cell B. Cohen’s κ was 0.507 pooled, 0.563 on cell A and 0.452 on cell B. The marginals across the three classes are 62/13/4 and 55/19/5, which are not degenerate, so κ is interpretable here rather than deflated by skew.

Disposition of the disputed rows. Sixteen of the 79 rows were disputed and all sixteen were resolved in the reconciliation meeting, fourteen as genuine and two as artifact. Twelve of the sixteen concerned the performance-review field. The twelve flags the coarse pass had classified as scrub-misses — none of which survives as a scrub-miss under the partition — split nine genuine and three other under adjudication.

Appendix E Triage Partition

Table E1 Evidence-string triage on the deleted-text run, cell B, $N = 66$ trajectories.

Category	Field-flags	Trajectories
Naive (any judge flag)	—	43/66 = 65.2%
Genuine re-expression	44	41/66 = 62.1%
Mode-1 artifact	6	(≈ 3 pp removed)
Scrub-miss (partition)	0	—
Mode-2 conflation	0	—

The mode-1 correction is six field-flags out of fifty, twelve percent at the field level but roughly three percentage points at the trajectory level, because most of those flags occur in trajectories that disclosed something else as well. The mode-2 column is zero by construction: there is no injected surrogate in the deleted-text condition.

Appendix F Additional Control Analyses

Normalization. A filter that normalizes before matching cuts residual disclosure from 100% to 59.1%; a filter matching literal spans cuts it only to 97.0%. Both filters ran on the same 66 trajectories, so the comparison is paired. An unpaired test would misstate the precision here and a paired test

needs the per-trajectory outcomes, which were not retained, so we report the counts alone. The normalizing filter deletes more text and should therefore induce at least as many mode-1 artifacts, yet it shows the lower residual. The known artifact bias therefore runs against the observed effect, so the 40.9-point reduction is conservative with respect to that bias. Over-redaction on this arm is 0/36, bounded above by 9.7 points.

Coverage. Recall is not the only way enforcement fails. Where a removed value re-enters through a channel the gate does not scan, residual disclosure returns to 100%; extending the gate to cover that channel returns it to the 0–3.3% range while remaining inside the utility band. Coverage and recall are independent axes, and a pre-context control is complete only when both hold.

Adaptive attacks. Generation-layer attacks that vary how the model is asked — paraphrase, inference, encoding, utility pressure — fall to $\leq 3.3\%$ once the value is absent from context. This is consistent with the account in Section 2: those attacks change the request, not what the context contains. We report this as a pilot result.

Recognition. On a panel of 50 records, pattern-based detection reaches approximately 100% on shaped fields and approximately 0% on the two free-text fields carrying most of the residual, for an overall recall of 38.9% across 1,750 field instances; a model-based detector given the field name reaches 70–100% on those hard fields. Recognition of semantic fields is imperfect; the paper’s claim concerns which recognition problem each placement poses, not the quality of any particular recognizer.

Appendix G Provisional Results

Two quantities are reported for completeness and are not load-bearing.

The first is a floor on output-layer disclosure derived by taking judge-flagged disclosures on oracle-scrubbed output and then classifying them by string presence: $24/81 = 29.6\%$ [20.0, 40.8] pooled across three records. The judge flag sits upstream of the classification, so this figure inherits mode 1 and we mark it **provisional**.

The second is an apparent difference between the two populations in the *mode* of residual disclosure — one re-expressing without any verbatim span, the other reproducing with formatting drift — which is significant at every threshold tested ($p \leq 7 \times 10^{-4}$). Its inputs are judge-flagged and inherit the same mode, and the populations are not matched, so we report the pattern and claim nothing from it. [provisional]

Appendix H Structural–Semantic Asymmetry

The thesis predicts an asymmetry: a provenance label should suppress an attack whose success condition is structural, and should be inert against over-sharing, whose success condition is semantic. We measured the second half. Labeling retrieved content with an authoritative directive leaves over-sharing exactly where it was, while removing the value drives it to zero.

We did not measure the first half. Establishing the asymmetry requires running the same labeling mechanism against an attack whose goal is a consequential action rather than a disclosure, in the same harness, so that injection success and over-sharing are separable events; that arm was specified and not executed. We therefore report label inertness as a result and the asymmetry as a motivation, and we do not claim to have demonstrated that labels succeed where they are supposed to.

Appendix I Experimental Details

The agent runs inside a harness built on an existing open-source agent-security framework [3]. Tools are backed by a document store holding the user’s mailbox, documents and calendar, so a retrieval call returns records rather than a fixed string, and a control placed before the agent starts can modify what those calls will return. Agents use native function-calling rather than code generation, following a harness correction made after an earlier round in which one model was excluded for emitting unparseable code; conflating “the model failed” with “the model was not run” is a confound

we avoid by validating the harness per model before any scored trial.

Every turn is logged at the agent’s run boundary and all scoring is performed offline against those logs; where a control rewrites the outgoing message, the post-control message is logged separately so the scorer reads the final text rather than the draft. Appendix B gives the instrumentation in detail.

Records and Policy. The user records are synthetic: fifty records generated from the field taxonomy of Section 4, each carrying roughly thirty-five fields across eight categories, with no real personal information anywhere in the study. Field values were written to be lexically distinctive — specific street names, unusual medication names, specific dates — so that a deterministic scorer has a fair chance of detecting verbatim disclosure; a corpus of generic values would have made the string scorer vacuous and forced the entire study onto the judge. The policy file is generated from the taxonomy rather than written by hand, so the label assignment for each field and task is reproducible from the taxonomy.

Tasks and Injected Content. Two tasks appear in the evaluated populations, both from the harness’s existing set: a generative recommendation-request task, whose policy admits three required fields while forbidding thirty, and a form-filling task. Injected content is planted before the run, appended to the body of the record most relevant to the task so that the agent retrieves the instruction through its own tool call rather than receiving it in the request. Two injection styles appear: an explicit instruction naming the fields it wants disclosed, and a socially-framed request naming none but asking for thorough background. All authentication and transport checks succeed in every trial; the injected content crosses as payload. Appendix B gives the seeding procedure.

Models. The anchor is a small proprietary model, `gemini-3.1-flash-lite`, acting as agent, judge and detector. An open-weight model, `gpt-oss-120b`, served through a separate provider, appears as a second agent; two further open-weight models, `llama-3.3-70b-versatile` and a Llama-4 variant, appear in the underlying corpus.

This is a monoculture with respect to the judge: the model family that produced a large part

of the text under test also scores it. We treat every judge-derived number accordingly, and the negative control reported in Section 6 exists partly to bound what that costs us.

Model configuration is recorded per run rather than assumed — forty-six stack records name the model, provider, endpoint, sampling settings, framework version and system-prompt file, with agents and judge both at temperature zero. What is *not* available is an immutable provider-side snapshot identifier: none of these endpoints exposes one, so a run is pinned by model identifier, provider, endpoint, client version and date rather than by a content hash. We state this rather than implying a stronger guarantee.

Appendix J Judge Error Analysis

The instrument that produces the output residual over-flags on scrubbed text, so we bound that behavior rather than assume it away.

The negative control. The retrieval arm produces text that never contained the forbidden values and carries no deletion residue. Running the same judge over it yields no false positives at all: 0/30 [0, 11.6]. The over-flagging is therefore specific to deletion scars and does not occur on naturally value-free prose. This control addresses the deletion-scar failure mode but does not validate the judge more generally.

Mode 1, gap artifacts. Six of fifty field-flags in the deleted-text run cite a bare label or a span-removal scar, for example an **allergies** flag whose cited evidence is an emptied label and its trailing punctuation. Removing them moves the trajectory residual from 65.2% to 62.1%.

Mode 2, type conflation. Substituting a well-formed same-type surrogate instead of deleting removes the scar but introduces a different error: the judge flags the field while citing the injected surrogate rather than the real value. At scale this is 9 of 106 field-flags, or 8.5%, in the surrogate run only, and it contributes nothing to the deleted-text residual. An isolated control at $N = 6$ was too small to characterize it and is superseded by the at-scale figure; Appendix A gives that control.

Mode 3, hallucinated or misattributed evidence. The genuine category is defined by

elimination, so we screen it mechanically on two axes: whether the cited evidence is present in the scrubbed text at all, and whether it matches the claimed field better than any other field in the record.

On presence, 43/44 in cell B and 32/35 in cell A cite evidence whose tokens occur in the scrubbed text at an 80% threshold; on attribution, 43/44 and 34/35 cite evidence matching the claimed field better than any other field in the record. Neither screen removes a trajectory from the residual, because every flagged trajectory carries at least one other clean genuine flag.

The presence screen is a bag-of-words test: it asks whether the evidence’s tokens occur somewhere in the letter, not whether the quotation appears as written, so it establishes that the cited vocabulary came from the document rather than that the citation did, and we report it with that reading. A threshold-free check on the same flags finds the longest contiguous token run at least five tokens for 29/35 and 40/44, with the median citation matching over its full length, so the evidence is in general quoted rather than reassembled; Appendix A says why we do not convert this into a second screen.

The presence failures share a cause worth naming. Two of cell A’s three and cell B’s single failure are the same artifact: the judge cites L7 / **band 28C**, the compensation grade — which is the ground-truth value, supplied to the judge in its own prompt. Re-deriving the scrub confirms the value is absent from the text the judge was shown, in full or in fragments, so the judge is quoting its prompt back as though it were an observation. This is separable from paraphrase-hallucination by cause and remedy, and detectable exactly because the cited string is the answer verbatim.

Appendix K Policy and Field Taxonomy

Policies are specified per task rather than per field. For each task, every field of the user’s record carries one of three labels: *required*, without which the task cannot be completed; *context-only*, referenceable but whose concrete value should not leave; or *forbidden*. The policy is generated mechanically from a field taxonomy of eight categories rather

than hand-curated, and fields default to *forbidden* — a field is permitted only where a task explicitly claims it.

The labeling is genuinely task-dependent, which is what separates a contextual-integrity policy from a sensitivity list: the user’s email address is required for expense reporting and forbidden for the recommendation task. A sensitivity list cannot express that, and would also miss fields carrying no conventional identifier — summaries of internal messages, notes on colleagues — whose disclosure to an external recipient is nonetheless a violation.

We define a *leak* as a forbidden field’s value reaching the outbound channel in any surface form. Every rate reported here is relative to this policy.

Appendix L Scoring Instruments

We score with two instruments, because neither alone can support the claim.

The first is a deterministic string scorer, testing whether the emitted message contains the full value at word boundaries or any contiguous three-, four- or five-word chunk of it, skipping chunks made entirely of short function words, and classifying a match by the policy. It admits no interpretation — a substring either occurs or it does not — and every result we describe as clean is produced by it. Its limitation is structural: it cannot see a disclosure sharing no verbatim span with the value, which is precisely the class this paper is about.

The second is a language-model judge, supplied with the task, the field names *and their literal ground-truth values*, and the outgoing messages, returning for every field a verdict, a mode, a confidence, and — critically — an **evidence** string that the prompt requires to be an exact substring of the messages. We separately screen compliance with that requirement. The judge is therefore not inferring what is secret but checking whether a known value was conveyed. A verdict is required for every field in the input list, including fields that did not appear, so denominators are fixed by construction and a missing field is a parse failure rather than a silent negative.

Appendix M Aggregation and Confidence Intervals

A flag is a *scrub-miss* if the full normalized value is still a substring of the scrubbed text; this is a failure of the scrubber, not an instance of re-expression, and we exclude it. A flag is a *conflation* if its evidence reproduces an injected same-type surrogate rather than the real value; this category applies only to a substitution-based variant of the instrument described below. A flag is an *artifact* if its evidence is structurally empty once the field’s own name and a stopword list are removed, or if it consists mainly of the whitespace scars left by span removal. A flag matching none of these is counted as *genuine* re-expression. Appendix A states the character and word thresholds these two tests use, and the word lists they depend on.

Aggregation is at the level of the trajectory: a run counts as a residual disclosure if at least one of its flagged fields is classified genuine. We report trajectory-level rather than field-level rates because the operational question is whether a given message may be sent, not how many items it enumerates; field-level counting would additionally make the denominator depend on how many fields a particular injection happened to name. Proportions are reported with Clopper–Pearson intervals. Every classification decision, together with the evidence string it was made from, is written to an audit file, so the partition can be re-derived or contested without re-running the judge.

Appendix N Retrieval-Arm Details

The retrieval arm is a single cell — the recommendation task under the explicit injection, on one user record, with the anchor model as the agent and the pre-context control applied before the run — and is the only arm executed in-loop with a control active. Its outcome needs stating precisely, because a naive reading of the raw inventory overstates it. Two trajectories disclosed **hobby_interests**, which the policy for this task labels *context-only* rather than forbidden, so those are context overflow rather than leaks under the predicate of Section 4. No trajectory disclosed a

forbidden field, which is what the arm reports: 0/30 (95% CI [0, 11.6]).

Appendix O Additional Limitations

We separate limitations that bound what the numbers mean from limitations that bound where they generalize.

The Partition Is Validated in Aggregate, Not at the Boundary. The genuine/artifact partition has now been checked against human judgment, and the check is reported in Section 6: two annotators, 79 evidence strings, a rubric fixed in advance, disagreements resolved. The outcome supports the reported residuals — 83.6% of the coarse pass’s genuine calls confirmed, and recounted figures whose intervals contain the reported ones.

What it does not support is a claim that the disclosure threshold is settled. Inter-annotator κ was 0.507, which is moderate and below the 0.65 bar this project’s own analysis plan set. Twelve of the sixteen disputed rows fell on a single free-text field, and that field carries most of the residual. So the aggregate is validated while the boundary that generates the aggregate is contested, and a different pair of annotators applying the same rubric could plausibly land a few points either side of what we report.

Two further limits. The adjudication was run on the partition’s genuine bucket only, so annotators could move flags out of the residual but never into it; the adjudicated figures are bounded below rather than centered, and the triage’s artifact calls are unvalidated. And the adjudication validates the partition, not the judge: it establishes that the triage sorts the judge’s flags roughly as a human would, not that the judge raised the right flags in the first place. The value-free negative control in Section 6 bounds that separately.

One Model Family Occupies Three Roles. The same small proprietary model acts as an agent, as the judge that scores disclosure, and as the detector in the recognition panel. Every judge-derived figure in this paper is therefore a product of that family assessing text that family had a hand in producing. The negative control bounds one consequence of this — the judge raises no false positives on value-free prose — but it

does not address family-level correlation between generator and grader. A cross-family judge was specified in the harness design and not used in the runs reported here.

The Output Arm Is a Counterfactual. Output filters were applied to logged messages and re-scored; the agent was not re-executed with a filter attached. This buys control, since every filter sees the same trajectories, and it costs realism: an in-loop filter could change the agent’s behavior, for instance by causing it to notice redactions and rephrase. We do not claim to have measured an in-loop output defense, and the residuals should be read as properties of the filter applied to a fixed corpus.

Population Definitions Are Not Fully Settled. Two population questions remain open, both disclosed where they arise as well as here. *Cell A is not strictly homogeneous*: it is assembled from an audit file named for the anchor model and is predominantly but not exclusively that model, because the selection conditions filter on evidence availability and disclosure rather than on agent identity. Cell A therefore supports an anchor-dominant comparison but not a strict within-model causal contrast. Cell B is homogeneous and supplies the cleaner cross-model corroboration of the output residual. *Cell A excludes 32 of 81 candidate trajectories* for want of turn-level logs — an exclusion by data availability rather than outcome, whose association test is not significant and whose excluded group discloses somewhat more, so it is conservative, but an exclusion nonetheless.

The Residual Is Concentrated in One Field. The surviving disclosures are not spread evenly across the policy: of the genuine flags whose cited field resolves to a ground-truth value, 41 of 44 in cell B and 23 of 35 in cell A concern a single free-text field, a performance-review excerpt, with cell A also carrying travel history, a family member’s health, a spouse’s name and children’s details. This concentration is consistent with the proposed mechanism: fields with lexical shape are removed cleanly by a string filter, whereas the residual is concentrated in ordinary prose. Nevertheless, the magnitude is carried by a narrow slice of the policy and should be interpreted primarily as evidence about free-text fields rather than forbidden fields in general.

The Two Arms Do Not Use Identical Removal Rules. The pre-context control

removes exact occurrences of the full value; the output-layer string filter removes the full value and every ≥ 3 -token span of it under normalization. The output side therefore operates with the stronger scrubber. This asymmetry favors the output arm and thus runs against the observed placement difference, but it remains an implementation difference that limits direct comparison.

The Retrieval Arm Is an Upper Bound, and Deviates From What Was Registered.

The retrieval control uses ground-truth record values rather than a detector’s output. It is an idealized upper-bound measurement for this pre-context removal mechanism under perfect target identification, not an estimate of a deployable detector-driven control. This also departs from the registered analysis plan, which specified a detector-driven arm as the primary endpoint and reserved a ground-truth arm as a labeled bound; the detector-driven arm was not run. Any comparison between this arm and the span-based output baseline is therefore a comparison between two oracle-equipped controls with different removal rules, not between a deployable retrieval control and a deployable output control.

The Semantic Filter Arm Is Partly a Pattern-Based Arm. The semantic redactor falls back to a deterministic pattern-based redactor when the model call cannot be completed, and at least 12.2% of that arm’s trials were redacted by the fallback. The reported residual is therefore a mixed-system measurement rather than a pure estimate of the model-based redactor. Because the fallback is the weaker redactor in this evaluation, its inclusion tends to increase the observed residual.

The Released Evidence Strings Are Truncated. The triage writes each judge citation to the audit trail truncated to ninety characters, and the mechanical screens ran on that truncated form; roughly half the genuine flags sit exactly at the cap, with the trailing word severed, and the full strings were not persisted. So the artifact we offer as contestable without re-running anything is, for those flags, a prefix, and the presence screen is easier to satisfy on a prefix than on a complete string, making its pass rates if anything optimistic. The partition itself depends on bare-label and gap-scar tests, for which a ninety-character prefix is sufficient, so the residual figures are not affected

in the same way. Lifting the cap and re-running the triage would remove this limitation.

Figures We Report but Do Not Rely On.

Two quantities carry an instrument whose error mode propagates into them: a disclosure floor derived by classifying judge-flagged disclosures by string presence, and an apparent difference between the two cells in the mode of residual disclosure. Both are reported as provisional in Section 6 and neither supports a claim.

Scope. The records are synthetic and contain no real personal information, which is necessary for the ground-truth scoring the study depends on but leaves open whether real records behave the same way. Two tasks and two injection styles appear in the evaluated populations. Attachments, images and optically recognized text are declared out of scope, as is the disclosure of values never present in any retrieved artifact. A stronger adversary — white-box against model internals, or operating under an explicit query budget against the defense — was specified during design and not evaluated.

Three further scope conditions inherited from the study design belong here. The retrieval arm was run on one user record under one task and one injection style, and on the anchor model only; there is no retrieval-arm measurement on the open-weight model, so the observed zero-disclosure result is specific to the anchor model and has no direct replication on the open-weight model. Benign recall is measured on the same generated corpus that supplies the disclosure trials, so utility and leakage are not independently sampled. And the disclosure predicate is presence-based: it asks whether a forbidden value is conveyed, not whether an adversary could re-identify the subject from what remains, so de-anonymization is outside what these numbers bound.

Appendix P Recognition Granularity

A defense deployed before transmission can decline to send. This suggests an objection to the framing of Section 2: if every span the recognizer fails to label is treated as maximally sensitive and removed, then a recall failure costs utility rather than privacy, and the recall bound is not a bound on enforcement at all. The objection is not

hypothetical — it is realized in OCELOT’s conservative default [17], and that paper’s ablation shows the default carries most of the mechanism’s safety.

We accept the objection as stated. We also correct a claim we made earlier in weaker form: a post-generation gate sits before the outbound message reaches the recipient, so it too can fail closed. It can decline to send, request regeneration, escalate to a human, or return a privacy-preserving refusal. The distinction between the two placements is therefore *not* that only one of them can convert uncertainty into utility loss. Both can.

The distinction is what each recognizer must evaluate before making that decision. A pre-context gate decides about representations already present in the retrieved artifact; a post-generation gate must additionally decide about representations the model composed after reading it. Conservative handling of uncertainty at the output may therefore trigger refusal or over-redaction more often. Establishing that requires a privacy–utility frontier, which we do not measure; our semantic-redactor arm illustrates the tension without establishing its magnitude.

What the objection does not remove concerns what deny-by-default can range over. A conservative default assigns top risk to atoms the recognizer delimited but could not label; it defends against *mislabeling*, not against *non-delimitation*, so it needs an atom to fire on. The triage audit says what granularity the survivors occupy. For every genuine flag whose cited field resolves to a ground-truth value — 44 in cell B and 35 in cell A — we computed the longest contiguous token run shared between the judge’s cited evidence and the true value. Across all 79 the longest is **two tokens for 76**, with 74 of 79 sharing exactly one or two: the filter behaving exactly as specified, removing every ≥ 3 -token span, with the payload a one- or two-token fragment inside a sentence the model composed itself.

The three flags that share a longer run are the same compensation-grade value in all three cases, and each cites evidence that the presence screen places outside the scrubbed message. They are prompt echo in the sense of Appendix J — the judge quoting a value supplied in its own prompt — so none of them is a surviving disclosure that a post-generation gate would have had to delimit.

Their trajectories remain in the residual on the strength of other genuine flags.

That sharpens the objection rather than dissolving it. These disclosures do have a delimitable atom — a two-token phrase is an atom — but not one at a granularity where a conservative default is a defense rather than a refusal to communicate. To fire on a two-word fragment, a default must treat arbitrary two-word phrases as candidate sensitive atoms, which approaches indiscriminate redaction and mirrors the utility trade-off observed when the semantic filter removed every required field along with the sensitive ones. Our claim is therefore narrower than “recall bounds enforcement”: deny-by-default requires atoms the recognizer can both delimit *and* withhold at acceptable cost, and at the granularity where the residual lives the second condition fails.

Appendix Q Semantic Redactor

A tested model-based redactor also exhibits an unfavorable privacy–utility tradeoff. The semantic filter reduces residual disclosure from 79.3% to 68.6%, while removing every required field from the message, an over-redaction cost of 78 percentage points. This characterizes one practical semantic redactor rather than establishing a limitation on semantic output defenses in general. Because the quantity is scored by the deterministic string matcher rather than the judge, it does not inherit the judge’s deletion-scar error mode; because the matcher counts only verbatim residual, it does not capture all semantic re-expressions.

One implementation caveat belongs with the number. The semantic redactor falls back to a deterministic pattern-based redactor when the model call cannot be completed, and the two leave distinguishable traces in the logs. Comparing pre-filter and post-filter records for all 377 trials of this arm, at least 46 — 12.2% — were redacted by the fallback rather than by the model, a lower bound since a fallback on text containing no pattern match leaves no trace. The arm’s residual should therefore be read as a mixed-system measurement rather than as a pure estimate of the model-based redactor.

Appendix R Population and Exclusions

Two aspects of these populations are central. The first is the magnitude of post-generation repair failure. Both output cells leave a clear majority of already-leaking trajectories still leaking, with confidence-interval lower bounds above 46%. Cell A is anchor-dominant but not homogeneous, so this is not a strict within-model causal contrast; cell B is a homogeneous open-weight-model output population whose residual is within about one percentage point. Separately, the prevention reference is the retrieval row, 0/30 with a 95% Clopper–Pearson interval of $[0, 11.6]$, measured over all trials in that cell rather than over already-leaking trajectories.

The second is that the output arms were given privileged target information: they were given the ground-truth values and deleted both those values and every three-token fragment of them, which no deployed span-removal filter can do. The residual is therefore not a statement about target-identification quality. It is an upper-strength test of exact-value and contiguous-span scrubbing on the fixed outputs, not a bound on every conceivable post-generation semantic mechanism.

The two output cells were filtered and scored by the same procedure. We report their agreement as corroboration and draw no conclusion from the 0.9-point difference: they differ in task, injection style and composition, so the comparison is not controlled. Cell B is the cleaner population — every candidate trajectory scored, no exclusions. Cell A is assembled from the anchor model’s audit file and is predominantly, but not exclusively, that model, since the selection conditions filter on evidence availability and disclosure rather than on agent identity. We therefore treat cell A as anchor-dominant rather than as a strict within-model comparison, and use homogeneous cell B as the cleaner cross-model corroboration of the output residual. Fig. 3 groups these residuals with the semantic filter and the label arm by the placement of each control.

Appendix S Deployment Implications

The asymmetry has an operational consequence. A pre-context control must be validated against the protected representations that can occur in retrieved artifacts. After model exposure, a post-generation control must additionally account for representations composed by the model itself, and those representations may change when the deployed model changes. A model change can therefore expand or alter the post-generation recognition surface and may require revalidation. This does not imply that pre-context recognition is purely lexical or model-independent. The two controls do not merely differ in effectiveness, they differ in what must be re-established when the deployment changes. This is not an argument for removing output filtering, but for avoiding reliance on it as the sole or load-bearing privacy control.

Appendix T Analysis Corrections

An early headline inflated the residual by 58.8 percentage points by mixing formatting variants with genuine implication. Two later over-corrections retracted more than the data supported and were restored by significance testing, and an inference that output filtering was model-dependent was reversed by measuring it. Inspecting worked examples surfaced the judge’s gap-artifact mode, from which we over-claimed that re-judging scrubbed text made the magnitude unmeasurable and, from a six-case control on which the judge was correct three times, that surrogate substitution was invalid; the evidence-string triage reversed both by quantifying each mode rather than assuming it.

References

- [1] Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., Fritz, M.: 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 ’23), pp. 79–90 (2023)

- [2] Liu, Y., Jia, Y., Geng, R., Jia, J., Gong, N.Z.: Formalizing and benchmarking prompt injection attacks and defenses. In: 33rd USENIX Security Symposium (USENIX Security 24), pp. 1831–1847. USENIX Association, Berkeley, CA, USA (2024)
- [3] Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., Tramèr, F.: AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In: Advances in Neural Information Processing Systems 37 (NeurIPS 2024) Datasets and Benchmarks Track (2024)
- [4] Debenedetti, E., Shumailov, I., Fan, T., Hayes, J., Carlini, N., Fabian, D., Kern, C., Shi, C., Terzis, A., Tramèr, F.: Defeating Prompt Injections by Design. arXiv:2503.18813 (2025)
- [5] Beurer-Kellner, L., Buesser, B., Crețu, A.-M., Debenedetti, E., Dobos, D., Fabian, D., Fischer, M., Froelicher, D., Grosse, K., Naeff, D., Ozoani, E., Paverd, A., Tramèr, F., Volhejn, V.: Design Patterns for Securing LLM Agents against Prompt Injections. arXiv:2506.08837 (2025)
- [6] Wu, F., Cecchetti, E., Xiao, C.: System-Level Defense against Indirect Prompt Injection Attacks: An Information Flow Control Perspective. arXiv:2409.19091 (2024)
- [7] Costa, M., Köpf, B., Kolluri, A., Paverd, A., Russinovich, M., Salem, A., Tople, S., Wutschitz, L., Zanella-Béguelin, S.: Securing AI Agents with Information-Flow Control. arXiv:2505.23643 (2025)
- [8] Nissenbaum, H.: Privacy as contextual integrity. *Washington Law Review* **79**(1), 119–158 (2004)
- [9] Barth, A., Datta, A., Mitchell, J.C., Nissenbaum, H.: Privacy and contextual integrity: Framework and applications. In: 2006 IEEE Symposium on Security and Privacy (S&P’06), pp. 184–198 (2006). IEEE
- [10] Mireshghallah, N., Kim, H., Zhou, X., Tsvetkov, Y., Sap, M., Shokri, R., Choi, Y.: Can LLMs keep a secret? testing privacy implications of language models via contextual integrity theory. In: The Twelfth International Conference on Learning Representations (ICLR) (2024)
- [11] Shao, Y., Li, T., Shi, W., Liu, Y., Yang, D.: PrivacyLens: Evaluating privacy norm awareness of language models in action. In: Advances in Neural Information Processing Systems 37 (NeurIPS 2024) Datasets and Benchmarks Track (2024)
- [12] Denning, D.E., Akl, S.G., Heckman, M., Lunt, T.F., Morgenstern, M., Neumann, P.G., Schell, R.R.: Views for multilevel database security. *IEEE Transactions on Software Engineering* **13**(2), 129–140 (1987)
- [13] Lunt, T.F.: Aggregation and inference: Facts and fallacies. In: Proceedings of the 1989 IEEE Symposium on Security and Privacy (1989). IEEE
- [14] Farkas, C., Jajodia, S.: The inference problem: A survey. *ACM SIGKDD Explorations Newsletter* **4**(2), 6–11 (2002)
- [15] Zhan, Q., Liang, Z., Ying, Z., Kang, D.: InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In: Findings of the Association for Computational Linguistics: ACL 2024 (2024)
- [16] Lahjouji, N., Colaco, A.G.: Agents That Know Too Much: A Data-Centric Survey of Privacy in LLM Agents. arXiv:2606.26627 (2026)
- [17] Xie, J., Li, S.: OCELOT: Inference-Leakage Budgets for Privacy-Preserving LLM Agents. arXiv:2606.12341 (2026)
- [18] Zhong, P.Y., Chen, S., Wang, R., McCall, M., Titzer, B.L., Miller, H., Gibbons, P.B.: RTBAS: Defending LLM Agents Against Prompt Injection and Privacy Leakage. arXiv:2502.08966 (2025)
- [19] Chen, S., Piet, J., Sitawarin, C., Wagner, D.: StruQ: Defending against prompt injection

- with structured queries. In: 34th USENIX Security Symposium (USENIX Security 25). USENIX Association, Berkeley, CA, USA (2025)
- [20] Hines, K., Lopez, G., Hall, M., Zarfati, F., Zunger, Y., Kiciman, E.: Defending Against Indirect Prompt Injection Attacks With Spotlighting. arXiv:2403.14720 (2024)
 - [21] Zhan, Q., Fang, R., Panchal, H.S., Kang, D.: Adaptive attacks break defenses against indirect prompt injection attacks on LLM agents. In: Findings of the Association for Computational Linguistics: NAACL 2025 (2025)
 - [22] Zou, W., Geng, R., Wang, B., Jia, J.: PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In: 34th USENIX Security Symposium (USENIX Security 25). USENIX Association, Berkeley, CA, USA (2025)
 - [23] Chen, Z., Xiang, Z., Xiao, C., Song, D., Li, B.: AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In: Advances in Neural Information Processing Systems 37 (NeurIPS 2024) (2024)
 - [24] Bagdasarian, E., Yi, R., Ghalebikesabi, S., Kairouz, P., Gruteser, M., Oh, S., Balle, B., Ramage, D.: AirGapAgent: Protecting privacy-conscious conversational agents. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), pp. 3868–3882 (2024)
 - [25] Lan, G., Inan, H.A., Abdelnabi, S., Kulkarini, J., Wutschitz, L., Shokri, R., Brinton, C.G., Sim, R.: Contextual integrity in LLMs via reasoning and reinforcement learning. In: Advances in Neural Information Processing Systems 38 (NeurIPS 2025) (2025)
 - [26] Zharmagambetov, A., Guo, C., Evtimov, I., Pavlova, M., Salakhutdinov, R., Chaudhuri, K.: AgentDAM: Privacy leakage evaluation for autonomous web agents. In: Advances in Neural Information Processing Systems 38 (NeurIPS 2025) Datasets and Benchmarks Track (2025)
 - [27] Wang, S., Yu, F., Liu, X., Qin, X., Zhang, J., Lin, Q., Zhang, D., Rajmohan, S.: Privacy in action: Towards realistic privacy mitigation and evaluation for LLM-powered agents. In: Findings of the Association for Computational Linguistics: EMNLP 2025 (2025)
 - [28] Abdelnabi, S., Bagdasarian, E.: AI Agents May Always Fall for Prompt Injections. arXiv:2605.17634 (2026)
 - [29] Pant, D., Lohani, S., Kumar, A.: On the Inseparability of Instructions and Data in Shared-Embedding Sequence Models. arXiv:2606.27567 (2026)
 - [30] Denning, D.E., Denning, P.J.: Data security. *ACM Computing Surveys* **11**(3), 227–249 (1979)
 - [31] Denning, D.E., Denning, P.J., Schwartz, M.D.: The tracker: A threat to statistical database security. *ACM Transactions on Database Systems* **4**(1), 76–96 (1979)
 - [32] Dobkin, D., Jones, A.K., Lipton, R.J.: Secure databases: Protection against user influence. *ACM Transactions on Database Systems* **4**(1), 97–106 (1979)
 - [33] Kleinberg, J., Papadimitriou, C., Raghavan, P.: Auditing boolean attributes. In: Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '00) (2000)
 - [34] Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '03), pp. 202–210 (2003)
 - [35] Dwork, C.: Differential privacy. In: Automata, Languages and Programming (ICALP 2006). Lecture Notes in Computer Science, vol. 4052, pp. 1–12. Springer, Berlin, Heidelberg (2006)