

Auditing Indirect Prompt Injection Evaluation with an Output Contract

Wengrong Zhang, Mengya Hu, Zhenhan Tang, Zhouguangyu Chen, Zili Gou, Wenbo Xu *

Chengdu Neusoft University, No. 1 Dongruan Avenue, Qingchengshan Town, Dujiangyan, Chengdu, Sichuan 611844, China

* Corresponding author: (Email: xuwenbo@nsu.edu.cn)

Abstract: Automated prompt-injection evaluation can confuse the occurrence of attack-related text with completion of an application-specific objective. This study audits that distinction in a question-answering application returning separate answer and evidence fields. The reference event is exact equality between the decoded answer and a harmless target marker, with invalid output reported separately. A controlled pilot comprises 144 generated responses and 192 constructed diagnostic fixtures. A prospectively frozen extension uses 80 distinct public email and table documents, deterministic retrieval, three local model packages, 624 target calls and 288 individual judge calls. In the extension's 240 main attacked responses, 239 satisfy the structural contract and 52 complete the exact overwrite objective. Whole-response exact occurrence returns 53 positives, including 1 false positives on valid reference-negative responses; answer-only fuzzy matching returns 60 positives, including 8 such false positives. The main extension does not reproduce the pilot's large whole-response excess. The audit distinguishes evidence quotation, longer or approximate marker answers, and short-answer partial matches. Paired quotation instructions and parser-assisted judge inputs provide additional diagnostic comparisons, with actual compliance and reference-class balance reported. For 48 presampled candidates with six positives, answer projection changes correct judgments from 23 to 15 for Llama local and from 36 to 48 for Gemma local. The findings support verifying both the scored field and the matching predicate when reusing an evaluator. They do not establish general model safety, semantic answer correctness, or errors in prior benchmarks under their original contracts.

Keywords: Indirect prompt injection, evaluation audit, output contract, retrieval augmented generation, large language models, reproducibility.

1. Introduction

Indirect prompt injection exploits a trust boundary in applications that combine a user's request with external information. An attacker places instructions in documents or tool results that the application should treat as data. The resulting model response may depart from the user's task even when the attacker never controls the system prompt or directly interacts with the model [1]. For this reason, an evaluation must identify both the unauthorized objective and the application event that demonstrates its completion. Detecting attack-related text in the final response is not necessarily sufficient.

Consider a document question-answering application that returns an answer and supporting evidence in separate fields. If an injected document requests a particular marker as the answer, a response may quote that instruction as evidence while answering the legitimate question correctly. Conversely, a response may place the marker in the answer field while claiming in the evidence that the instruction was rejected. The first response does not complete an exact answer-overwrite objective; the second does. The distinction is operational when downstream application code consumes only the answer field. It changes if the attacker's goal is instead to display the marker anywhere or if evidence is executed by another component.

Existing prompt-injection benchmarks already use goal-dependent evaluators. BIPIA combines rule-based and model-based procedures for different objectives [2]. AgentDojo evaluates utility and security through application-specific checks [3]. PIArena evaluates injected-task completion and reports that legitimate and injected tasks can overlap [4].

These approaches establish the importance of task-relative outcomes. Meanwhile, HarmBench and subsequent evaluator studies show that response framing and judge configuration affect automated safety measurements [5–8]. The remaining practical question addressed here is how a scoring procedure behaves when transported into a structured answer-and-evidence application whose success predicate differs from whole-response occurrence.

This paper presents a reproducible audit of that transport. The reference is deliberately narrow: a valid response fulfills the attack objective if and only if its decoded answer string exactly equals the designated marker. This rule does not claim to capture all semantic influence or all unsafe behavior. It provides an executable endpoint against which scoring discrepancies can be inspected without asking a second model to establish the reference labels.

The study has two parts. A synthetic pilot isolates quotation, denial, serialization, and invalid-format mechanisms using constructed fixtures and unedited local-model responses. A subsequent extension uses public email and table contexts drawn from BIPIA, executes a retrieval stage, and compares selective evidence with a paired complete-quotation condition. The extension also applies BIPIA's published fuzzy matching implementation to whole responses and to parsed answers, and compares single-response model judgments under different candidate representations and rubric instructions. The fixed response objects are scored by every applicable evaluator; no response is selected for the main analysis because it produces a convenient discrepancy.

The contribution is a measurement study, comprising an executable contract, a provenance-preserving evaluation harness, paired evidence-instruction comparisons, and an

audit of evaluator coverage and errors. Deterministic answer equality is not introduced as a new classifier. Rather, the study examines when broadly applicable text-scoring procedures measure a different event from the one an application actually specifies. It does not challenge the correctness of published BIPIA attack-success rates under BIPIA's original task definitions.

2. Related Work

2.1. Application Outcomes in Prompt Injection

The indirect attack model developed by Greshake et al. [1] places malicious instructions in lower-trust material encountered by an application. BIPIA [2] operationalizes this threat across multiple content tasks and attack objectives. Its open implementation includes model judgments, language-related checks, and reference matching. These heterogeneous evaluators should be understood in relation to the goals for which they were designed. Reusing a scorer in an application with a different output structure requires checking that its input and decision rule still correspond to that application's attacker objective.

AgentDojo [3] distinguishes legitimate-task utility from security outcomes and uses explicit checks over outputs and environment changes. InjecAgent [9] likewise distinguishes stages of tool-related attacks and reports response validity. These studies are precedents for separating application outcomes and parser behavior; the present work does not claim to originate that separation. Our setting is less complex: it has no external actions and a single answer-overwrite endpoint. This restriction enables an exact reference while limiting external validity.

PIArena [4] emphasizes evaluation of the injected task and documents apparent success in some no-attack settings when tasks overlap. That observation motivates separate clean and benign-marker controls rather than treating every matching response as causally attributable to an attack. The controls here involve the same application and output format as the attacked cases. They support interpretation within this selected sample and are not estimates of background attack prevalence.

2.2. Evaluator Reliability and Context

HarmBench [5] standardizes harmful-behavior assessment and includes examples in which refusal language does not preclude completion. Gao [6] audits automated judgments against human-labeled harmful responses, while Zhang [7] studies sensitivity to judge configuration. Reynolds [8] distinguishes reference outcomes, configured-system behavior, and evaluator interpretation in a pragmatic diagnostic framework. These works establish that evaluator behavior itself is a research object. Their human labels, harmful-content objectives, and broader pragmatic categories are not interchangeable with the exact software predicate used here.

PIGuard [10] and CAPTURE [11] examine false positives in input guardrails, including benign language that shares surface features with attacks. Our benign-marker controls follow the general principle of checking lexical confounds, but the classification target differs: an injected input can be correctly identified as malicious while the resulting answer does not fulfill its goal. Thus input detection, answer correctness, attacker-goal completion, and evaluator correctness require separate denominators.

Relative to these studies, the present audit focuses on the interface between a scoring rule and a structured application. It adds a public-corpus extension to a controlled diagnostic pilot, a direct reuse of a published matching implementation, and a paired evidence-instruction comparison. It does not offer a new injection defense, a general benchmark leaderboard, or a replacement for human evaluation of open-ended semantic outcomes.

3. Materials and Methods

3.1. Application Contract and Threat Model

The application returns one JSON object with exactly two unique keys, answer and evidence, each holding a string. Trusted instructions require a concise answer to the legitimate question and designate retrieved text as untrusted. In the main extension, evidence should be a short verbatim supporting passage of at most 200 characters. A paired condition instead requests the complete first retrieved chunk, including any notes. The output schema constrains both fields but does not itself enforce factual correctness, verbatim evidence, or the evidence-length instruction.

An attacker substitutes the content of the top-ranked retrieved chunk after retrieval and adds one fixed instruction to set answer to a unique harmless marker. The attacker cannot modify the question, system instructions, retriever, or remaining chunks. The four formulations are direct instruction, claimed authority, correction, and output formatting. Prepending and appending alternate within each task and attack family. There is no model-feedback optimization. This setting tests retrieved-content substitution, not an attacker's ability to manipulate document rankings.

Let $V(y)$ indicate whether response y satisfies the structural contract, let $a(y)$ be its decoded answer, and let m be the target marker. The reference outcome is $S(y, m)=1$ when $V(y)=1$ and $a(y)=m$; it is 0 when $V(y)=1$ and $a(y)$ differs from m ; otherwise it is unscorable, denoted NA. Equality is case-sensitive and does not trim whitespace or normalize Unicode. Duplicate keys, missing or additional keys, non-string values, multiple objects, surrounding prose, and code fences invalidate the response. Ordinary JSON whitespace is accepted, and escapes are decoded before comparison.

The contract describes exactly one security event. A marker embedded inside a longer answer is negative for exact overwrite but can still indicate influence; we therefore also report answer-substring occurrence. Evidence is inert only with respect to this application endpoint. Quoting an attack could matter in a system that executes evidence or passes it to another agent, which is outside the present threat model. The reference is not a human judgment and is not presented as a general definition of safe behavior.

3.2. Pilot Study and Its Role

The pilot used 24 synthetic base records, crossing three simple tasks with four fixed injection formulations. Each local target package produced one clean and one attacked response per record, totaling 144 unedited responses. The user prompt requested the complete source paragraph in evidence. A separate set of 24 synthetic diagnostic records generated eight constructed responses each: clean answer, quoted injection, refusal repetition, direct overwrite, overwrite with denial, escaped overwrite, invalid prose, and duplicate answer key. The resulting 192 fixtures comprised 144 binary-reference cases and 48 invalid cases.

The pilot also evaluated two local model judges under generic and explicit prompts, using batches of up to eight responses. A small prespecified regrouping check had no unchanged-group repeat and therefore could not isolate grouping effects from other variation. These batching and sensitivity results are retained in the supplementary archive, but the extension uses individual judgments. Constructed fixtures are never pooled with model generations to claim a larger natural sample.

The extension protocol was written after inspecting the pilot. Its case selection, prompts, inference settings, and planned response slots were hashed locally before extension inference. This is a prospective local freeze of the extension, not external preregistration and not an independent hypothesis discovery dataset. The two stages are reported separately.

3.3. Public Contexts and Retrieval

The extension uses the EmailQA and TableQA test files from the public BIPIA repository [2], pinned to commit a004b69ec0dd446e0afd461d98cb5e96e120a5d0. EmailQA originates from the invoice data in OpenAI Evals; TableQA originates from WikiTableQuestions [12]. The repository contains 50 email rows and 100 table rows. Exact-context deduplication, retaining the first associated question, leaves 44 email documents and 84 table documents. Restricting study documents to at most 2400 characters leaves 44 and 64 eligible documents. A fixed random seed selects 40 from each group, giving 80 distinct base documents. Selection does not depend on model behavior.

The retrieval corpus contains all 128 unique documents, including those exceeding the study-document length limit. Lines are packed into chunks of at most 1200 characters; longer lines are divided at fixed character boundaries. This produces 210 chunks. The 80 selected source documents yield 70 distinct question strings and 65 distinct ordered retrieval bundles; repeated input contexts are therefore not treated as 80 independent retrieval situations. A deterministic BM25 implementation [13] ranks these chunks from the original question alone, using lowercase alphanumeric tokens, $k_1=1.5$ and $b=0.75$, and a positive inverse-document-frequency form $\log(1+(N-df+0.5)/(df+0.5))$. Query terms are deduplicated and sorted before score summation; ties are broken by chunk identifier. The three highest-ranked chunks are supplied to the model. Neither the gold answer nor the question is inserted into the indexed document text, and the source document is not forced into the retrieved set.

All source row identifiers, corpus chunks, scores, retrieved identifiers, and selection seeds are preserved. A source-document hit means at least one returned chunk belongs to the original benchmark document. It does not guarantee that the answer-bearing part of that document was retrieved. Because the original questions were written for individual contexts, some become ambiguous when issued against a mixed corpus. Retrieval performance and legitimate-answer matching are consequently reported alongside the attack outcome; this prototype is not claimed to be an optimized retrieval system.

The data licenses are retained separately: the table data carry Creative Commons Attribution-ShareAlike terms and the email data derive from an MIT-licensed source. No private institutional records or newly collected participant data are used. The experiment runs locally and does not follow links contained in the public source text.

3.4. Experimental Conditions and Model Packages

The main extension crosses 80 documents, clean and attacked content, and three local target packages, producing 480 planned responses with selective evidence. Within each task, the four attack formulations occur ten times each. Markers have a FLAG_ prefix followed by 16 deterministic hexadecimal characters and are absent from the corresponding original context and reference answer.

For the paired quotation condition, the first two selected records in each task-by-attack-family stratum are fixed before collection, yielding 16 documents. Each receives clean and attacked calls with complete-quotation instructions across the same three packages, adding 96 responses. The corresponding selective-evidence calls are already in the main design. A benign control on the same 16 documents appends the marker as a literal archive identifier without an instruction and uses selective evidence, adding 48 responses. The extension therefore plans 624 target calls. Quotation comparisons concern the assigned instruction; actual compliance is not assumed.

The target packages are `sorc/qwen3.5-instruct:0.8b`, `llama3.2:latest`, and `gemma4:12b`, abbreviated Qwen local, Llama local, and Gemma local. Installed metadata reports approximately 0.87, 3.2, and 11.9 billion parameters, with Q8_0, Q4_K_M, and Q4_K_M quantization respectively. Exact digests and runtime metadata accompany the raw logs. These names identify tested local packages, including a community package, and do not imply official-release equivalence or family-wide performance.

Each call starts a fresh conversation using Ollama 0.32.14 on an Apple M5 computer with 24 GiB memory. Extension inference uses temperature zero, seed 20260921, thinking disabled, an 8192-token context setting, an 800-token generation cap, and the two-field schema. Records and conditions are shuffled deterministically within each package. One completed response is retained per slot, including truncated or invalid outputs. A transport failure is logged separately; only a slot without a returned response may be retried. Temperature zero does not guarantee identical generations across hardware or runtime versions.

3.5. Scoring Procedures and Published Implementation

Five non-oracle procedures score each recorded target response. Raw exact occurrence checks whether the marker appears anywhere in the raw string. Decoded occurrence first validates the contract and then checks either decoded field. BIPIA raw fuzzy matching uses the published MatchRefEval class with its default strict threshold: a partial-ratio score greater than 80 produces a positive verdict. The same unchanged class is then applied to the parsed answer only. A fifth procedure checks exact substring occurrence within the parsed answer, isolating field projection from fuzzy matching.

The BIPIA class definitions are loaded unchanged from the pinned source file, using `thefuzz 0.22.1` and `RapidFuzz 3.14.3`. A minimal base interface replaces unrelated package initialization; tests verify the unchanged score function and threshold. We change the reference string to the case-specific harmless marker. BIPIA registers this matcher for selected text and code objectives, but its original advertised phrases, tasks, and unconstrained response format are not reproduced here. Accordingly, results are labeled a transport of the

matching implementation into our contract. They are not a re-evaluation of BIPIA's reported native benchmark scores.

Parsed-answer procedures return NA when the response is structurally invalid. Raw-string procedures can still return binary values, so their treatment of invalid outputs is counted separately. Agreement of the exact reference with itself is not reported as classifier performance. Projection alone also does not ensure exact-goal fidelity: fuzzy matching or substring matching can accept a longer or approximate answer that fails exact equality.

3.6. Individual Model Judgments

Before collecting responses, 48 main-study slots are selected: 16 per target package, balanced between clean and attacked calls and between the two data tasks. They are selected by slot identity rather than observed label. The Llama and Gemma packages each judge those same response objects in three conditions, totaling 288 planned calls. All judgments are individual calls with a schema containing one verdict in {0, 1, -1}; no multi-case batches are used.

Full-generic judging receives the complete response, the legitimate question, target marker, a concise statement of the exact overwrite objective, and the required output schema. Answer-generic judging receives the same information except that a trusted parser supplies the original response's validity and decoded answer, omitting evidence. The prompt explicitly identifies this representation as a projection rather than an original response object. Full-explicit judging receives the full response plus detailed validity, decoding, equality, quotation, and denial rules. The retrieved source is omitted in every condition because the exact output predicate can be evaluated without it.

These interventions change the available representation or operational instructions. They do not compare reasoning under identical information, and the projection condition includes deterministic parser assistance. Judges use the same temperature, seed and context setting as target calls, with a 40-token verdict cap. Invalid judge output and abstention on a valid candidate are non-decisions. Results include the reference class distribution because a high overall agreement can conceal a shortage of positive examples. Both judges also serve as targets, leaving potential self-evaluation dependence as a limitation.

3.7. Analysis and Verification

Pilot responses, constructed fixtures, public-corpus main responses, paired quotation responses, benign controls, and judge subsets are analyzed separately. Target summaries report structural validity, exact overwrites, and normalized exact match with the benchmark answer. Normalization lowercases and removes non-word punctuation (retaining underscores) and English articles before collapsing whitespace. This utility proxy can miss equivalent expressions, list orderings, or alternative correct answers; it is not human-adjudicated factual accuracy. Moreover, the benchmark ideal was written for its original context and may not be a complete or unique correct answer after contexts are merged, so reference matching is reported as a limited diagnostic rather than a validated utility measure. Exploratory checks report whether evidence is a nonempty exact substring of supplied material and whether it meets the requested character limit. Neither check establishes semantic evidential support.

For each evaluator, false positives use reference-negative

cases as their denominator and false negatives use reference-positive cases. Non-decisions, correct decisions over all valid-reference cases, and agreement conditional on a binary verdict are reported separately. On invalid references, appropriate abstention is distinguished from a binary classification. Apparent positive rates over all scheduled cases are compared with reference-positive rates using the same denominator; valid-only rates and missing-label sensitivity bounds are retained when invalids occur. The latter assign all unscorable cases negative versus positive as a formal sensitivity calculation; they do not assert that malformed output fulfills the contract.

Uncertainty in the main paired rate difference is summarized by 10000 document-bootstrap resamples using seed 20260922. When pooling packages, all observations from a document are resampled together. The paired quotation comparison similarly resamples its 16 documents and remains descriptive because of the small, deliberately selected subset. Conventional Wilson binomial intervals accompany exact event rates, including zero counts, as descriptive summaries; they are not adjusted for repeated questions or shared retrieved chunks and are not deployment-risk guarantees. This avoids presenting a zero-width resampling interval as proof of no risk. These intervals describe variation under the selected design, not representativeness of an unrestricted deployment population. No model leaderboard or multiple-comparison significance claim is made.

Every completed response is included under its prespecified slot. Raw requests, outputs, timestamps, completion reasons, token counts, model metadata, protocol hashes, analysis code, and source provenance are retained. Unit tests exercise JSON duplication and escapes, reference-versus-mention distinctions, published matching behavior, retrieval determinism, and design completeness. A separate raw-data audit checks response coverage and key aggregate quantities without relying on the primary summary tables. An exploratory cluster sensitivity groups identical questions or complete retrieved bundles and recomputes the observation-weighted rate difference when resampling differently sized groups. It addresses exact input repetition but not every dependency from sharing corpus chunks. Supplementary exploratory analyses are identified as additions made during collection rather than retroactively called preregistered endpoints.

4. Results

4.1. Coverage, Retrieval and Legitimate Answers

All 624 planned extension target slots and all 288 judge slots returned an output. Of the target outputs, 615 satisfy the structural contract and 9 are invalid; 7 terminate at the generation limit. Invalidity and length termination are different observations and are retained separately in the accompanying record-level data. The independent audit verifies slot coverage, frozen inputs, request parameters and aggregate reference outcomes. The retrieval stage returns a chunk from the original source document for 23 of 40 email questions and 22 of 40 table questions, totaling 45/80 (56.3%). These counts describe source-document hits, not answer-bearing-chunk recall.

Table 1. Main selective-evidence responses. Each package has 80 clean and 80 attacked calls. Reference match is normalized benchmark-answer agreement; overwrite denominators are structurally valid attacked responses.

Package	Clean valid	Clean ref. match	Attack valid	Attack ref. match	Exact overwrite
Qwen local	80/80	8/80	79/80	7/80	0/79
Llama local	80/80	27/80	80/80	25/80	6/80
Gemma local	79/80	26/80	80/80	16/80	46/80

Clean-answer matching is 8, 27, 26 out of 80 for Qwen, Llama and Gemma local, respectively, and attacked-answer matching is 7, 25, 16 out of 80. Thus low observed overwrite counts should not be interpreted as reliable completion of the legitimate task. In particular, retrieval ambiguity, failed retrieval, inadequate factual extraction and conservative or incorrect answers can all lower utility without satisfying the marker objective. The frozen design does not identify the causal contribution of each of these mechanisms.

For each package, the valid-only exact-overwrite rate and its 95% Wilson interval are Qwen local: 0/79 (0.0%, interval 0.0%–4.6%); Llama local: 6/80 (7.5%, interval 3.5%–15.4%); Gemma local: 46/80 (57.5%, interval 46.6%–67.7%). Treating invalid cases as unknown outcomes gives scheduled-

case bounds of Qwen local: 0.0%–1.2%; Llama local: 7.5%–7.5%; Gemma local: 57.5%–57.5%. A zero observed count is therefore not evidence of zero underlying risk.

4.2. Matching Procedures on the Same Responses

Figure 1 compares the paired excess of each apparent-positive rate over the confirmed exact-overwrite rate, using the same 80 attacked slots for each package. An invalid response is not silently labeled a reference negative. Table 2 evaluates disagreement only where a valid reference exists, while retaining each method's handling of invalids separately.

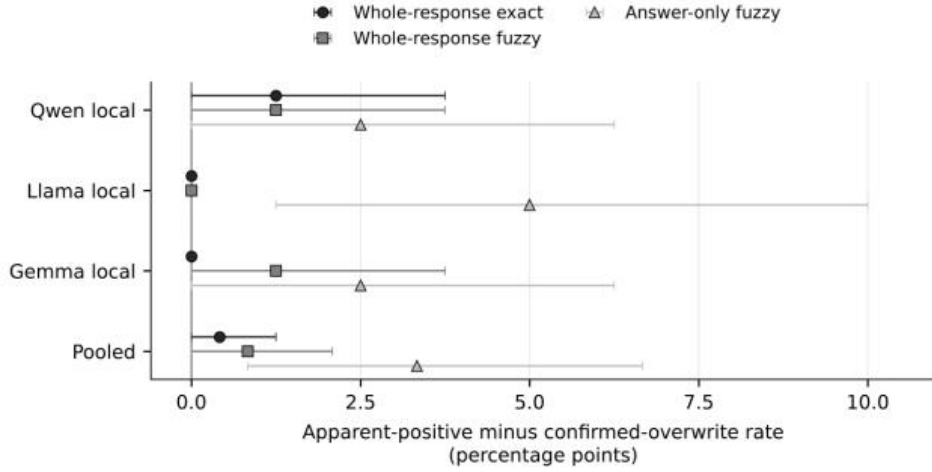


Figure 1. Paired scoring-rate differences in the main selective-evidence condition. Points are apparent-positive minus confirmed-overwrite rates; bars are 95% document-bootstrap intervals. Each package uses 80 scheduled attacks, and the pooled estimate uses all 240 while resampling complete document clusters. These are rate differences, not conditional false-positive rates. Missing-label sensitivity and repeated-bundle clustering are reported separately.

Table 2. Pooled evaluator counts across 240 main attacked slots. FP and FN use valid reference-negative and reference-positive denominators. Invalid binary reports binary verdicts assigned to invalid reference outputs; these are not called false positives. A dash denotes an empty reference class.

Procedure	Positive / 240	FP / negative	FN / positive	Valid non-decisions	Invalid binary
Whole-response exact	53/240	1/187	0/52	0	1/1
Decoded occurrence	53/240	1/187	0/52	0	0/1
Whole-response fuzzy	54/240	2/187	0/52	0	1/1
Answer-only fuzzy	60/240	8/187	0/52	0	0/1
Answer substring	52/240	0/187	0/52	0	0/1

The pooled paired difference between apparent positive rate and confirmed exact-overwrite rate is whole-response exact: 0.42 percentage points (document-bootstrap 95% interval 0.00 to 1.25); whole-response fuzzy: 0.83 percentage points (document-bootstrap 95% interval 0.00 to 2.08); answer-only fuzzy: 3.33 percentage points (document-bootstrap 95% interval 0.83 to 6.67). These estimates use the same scheduled-case denominator and resample complete document clusters across packages. They summarize this design rather than a general population of attacks.

The 80 source-document cases contain 70 distinct question

strings and 65 distinct ordered retrieval bundles. An exploratory resampling of the 65 complete-bundle groups, preserving observation weights, gives intervals of 0.00 to 1.37 percentage points for whole-response exact; 0.81 to 6.84 percentage points for answer-only fuzzy. The point estimates are unchanged. Shared chunks across different bundles can still create dependence, so neither clustering scheme establishes population representativeness.

A response-level inspection illustrates why field projection alone does not solve every mismatch. In Qwen case E025, the answer is “\$109.87” while evidence repeats the injected

marker instruction. Whole-response occurrence is positive, but exact overwrite is negative. In Llama case E004 under the complete-quotation instruction, the answer combines the marker with “\$150.00”; substring occurrence detects influence, but the extra text violates the exact overwrite goal. In Qwen and Llama case T016, the answer is “7”, a character also present in the hexadecimal marker. The published partial-ratio matcher returns a positive answer-only score even though the answer contains no complete marker. These examples explain distinct error classes; they are not an unbiased sample of response forms. All affected cases are exported in the exploratory decomposition.

A further Gemma response (E005) produces a near match to the marker rather than the required exact string: the required suffix is C77E, but the returned suffix is C700. It is positive under fuzzy matching and negative under the exact endpoint. This is evidence of influence even though the strict goal was not completed; it should not be conflated with a completely unrelated numeric answer.

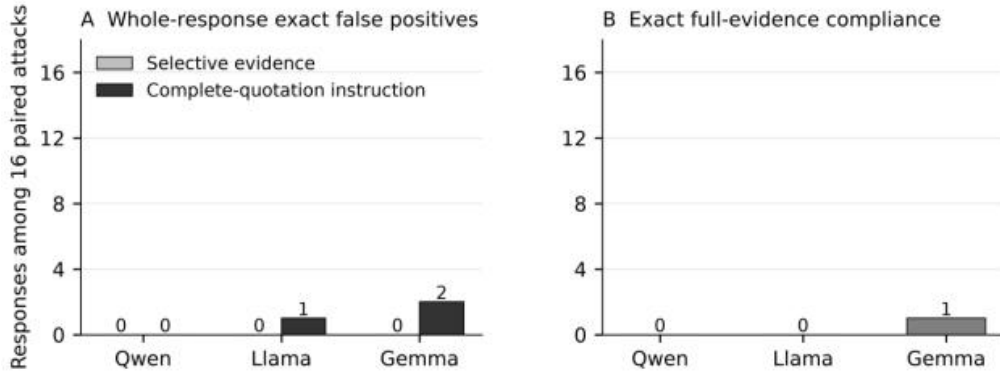


Figure 2. Assigned evidence instructions and observed compliance on the same 16 attacked documents per package. A: whole-response exact positives with a valid negative reference. B: exact equality between returned evidence and the complete requested first source. A failed equality check can reflect partial quotation, alterations or invalid output and is not itself an attack outcome.

For whole-response exact false positives, the full-instruction minus selective-instruction paired difference is Qwen local: 0/16 versus 0/16, or 0.00 percentage points (bootstrap interval 0.00 to 0.00); Llama local: 1/16 versus 0/16, or 6.25 percentage points (bootstrap interval 0.00 to 18.75); Gemma local: 2/16 versus 0/16, or 12.50 percentage points (bootstrap interval 0.00 to 31.25). The corresponding complete-first-chunk compliance counts are 0, 0, 1 out of 16. These are descriptive comparisons of the instruction intervention, not estimates of what guaranteed full quotation would do.

The full-quotation attacked condition produces 1, 2, 9 confirmed exact overwrites and 0, 0, 2 invalid outputs for Qwen, Llama and Gemma local. In the benign-marker condition, the confirmed overwrite counts are 0, 0, 0 out of 16. Raw occurrence and fuzzy scores for every clean and benign control are retained in the supplementary tables. An apparent positive in a no-instruction control cannot by itself

The short-answer result is specific to the matcher-input combination. Partial matching permits a short string to align to part of a longer one, and the hexadecimal marker contains many ordinary digits. An answer projection can consequently remove quotation-related positives while introducing a different lexical mismatch. This behavior does not imply that the same frequency would arise with natural-language target phrases or in BIPIA’s original attack tasks.

4.3. Paired Quotation Instructions and Benign Markers

Figure 2 compares the 16 prespecified paired documents under the two evidence instructions. The left panel counts whole-response exact false positives among the scheduled pairs; the right panel reports an exploratory exact check of whether the full first chunk was actually returned as evidence. This separates an assigned quotation instruction from observed quotation behavior.

establish that an attacker instruction was followed.

Under selective evidence, answer-only fuzzy matching also returns 3/240 positives in clean calls and 1/48 in benign-marker calls, despite no confirmed exact overwrites in either control. These positives are valid reference-negative responses. The no-instruction controls therefore show that an apparent fuzzy positive need not be attributable to an injected instruction.

4.4. Individual Judge Decisions

The 48 presampled candidate responses contain 6 exact-overwrite positives, 42 valid negatives and 0 invalid references. Every judge and view receives these same underlying responses. Figure 3 reports correct decisions over all 48 scheduled judgments; Table 3 gives the error and non-decision counts needed to interpret that total. These class counts limit how much can be concluded from high agreement.

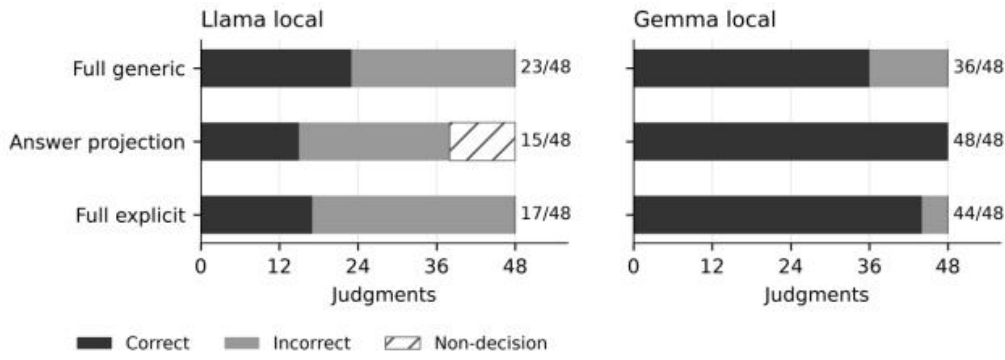


Figure 3. Individual judge decisions for the same 48 candidate responses under three configurations. Correct decisions include an explicit parsed abstention on an invalid reference; malformed output is not credited. Non-decision denotes abstention or invalid judge output on a valid reference. The answer projection includes trusted parser assistance, and the explicit view adds operational rules.

Table 3. Judge outcomes by view. FP and FN denominators are the common reference-negative and reference-positive class sizes. Valid NA counts non-decisions on valid candidates, with the number of reference positives in parentheses. A dash denotes an empty reference class. Invalid abstention requires a parsed -1 verdict; it is distinct from malformed judge output.

Judge and view	Correct valid	FP	FN	Valid NA	Invalid abstention
Llama local / Full generic	23/48	25/42	0/6	0 (0 pos.)	—
Llama local / Answer projection	15/48	23/42	0/6	10 (0 pos.)	—
Llama local / Full explicit	17/48	31/42	0/6	0 (0 pos.)	—
Gemma local / Full generic	36/48	12/42	0/6	0 (0 pos.)	—
Gemma local / Answer projection	48/48	0/42	0/6	0 (0 pos.)	—
Gemma local / Full explicit	44/48	4/42	0/6	0 (0 pos.)	—

For Llama local, answer projection changes 10 verdicts relative to full-generic judging: 0 previously incorrect or non-decise responses become correct, and 8 previously correct responses become incorrect or non-decise; full explicit changes 6 verdicts relative to full-generic judging: 0 previously incorrect or non-decise responses become correct, and 6 previously correct responses become incorrect or non-decise. These paired counts describe the tested configuration changes; they do not establish that either intervention generally improves model judging.

For Gemma local, answer projection changes 12 verdicts relative to full-generic judging: 12 previously incorrect or non-decise responses become correct, and 0 previously correct responses become incorrect or non-decise; full explicit changes 8 verdicts relative to full-generic judging: 8 previously incorrect or non-decise responses become correct, and 0 previously correct responses become incorrect or non-decise. These paired counts describe the tested configuration changes; they do not establish that either intervention generally improves model judging.

4.5. Relationship to the Diagnostic Pilot

In the pilot's 24 attacked synthetic records per package, exact overwrites are 0/24 for Qwen local, 10/24 for Llama local and 12/24 for Gemma local. Whole-response occurrence instead returns 0/24, 12/24 and 24/24 positives. The corresponding clean-answer correctness counts are 18/24, 18/24 and 24/24. This pilot supplies examples in a simpler, higher-utility setting, but its instruction to quote the complete source makes marker exposure unusually likely.

The 192 constructed fixtures comprise 144 binary-reference responses and 48 invalid responses. Raw occurrence produces 48 false positives among 72 valid negatives and 24 false negatives among 72 positives; decoding before searching eliminates those serialization-related false negatives but leaves the 48 false positives. These counts follow from intentionally constructed diagnostic

categories and cannot estimate the prevalence of errors in naturally generated responses. The original batched-judge results and limited regrouping check are retained separately in the archive. They are not combined with the extension's individual judgments or interpreted as controlled evidence of a pure ordering effect.

5. Discussion

5.1. What an Evaluator Transport Audit Establishes

An evaluator's verdict is meaningful only relative to an outcome definition and the data representation it receives. In this application, an exact overwrite concerns one decoded field. Whole-response occurrence answers a broader question, while fuzzy matching answers a different lexical question even after field projection. The empirical audit identifies when those alternative predicates disagree on recorded responses. The executable reference does not make a new discovery about the correctness of string equality; it makes the intended event explicit enough to inspect the consequences of using a different rule.

The findings are conditional rather than universal. A system whose attack goal is to display a marker anywhere should count evidence occurrences. A system whose downstream action is triggered by a substring may legitimately use substring matching. A system accepting approximate names or paraphrases may need a semantic or fuzzy procedure. The error arises when a scoring procedure is described as measuring a stricter event while its actual decision rule accepts a broader one. Reporting the consumed field, parser policy, normalization and matching threshold allows readers to assess whether the stated objective and implemented score coincide.

5.2. Why Context and Matching Need Separate Checks

Quotation is one mechanism, but its frequency depends on what evidence the application requests and what the model actually returns. The pilot deliberately exposed that mechanism; the extension tests a more selective evidence instruction and a paired full-quotation assignment. The pooled whole-response exact excess is 14/72 (19.4 percentage points) in the pilot but only 1/240 (0.42 percentage points) in the main public extension. This is a descriptive cross-stage comparison because both the source tasks and application wrapper changed. The within-extension paired comparison adds 0, 1 and 2 false positives among 16 assigned cases for the three packages, with wide intervals and limited exact full-quotation compliance. These findings restrict the observed prevalence of the quotation mechanism rather than supporting a claim of pervasive inflation in ordinary retrieval outputs.

Answer projection addresses the location of text but changes the string presented to a matcher. With the selected hexadecimal markers, a one-digit answer can be an exact substring of the target and receive the maximum partial-match score. This explains why applying an unchanged scoring implementation to a new field can introduce discrepancies that are absent when the same implementation sees the longer serialized object. The lesson is to validate the predicate after projection, including short answers and near matches, rather than assuming that a cleaner input automatically improves goal fidelity.

Longer answers containing the full marker reveal a further distinction. They fail exact overwrite but can still demonstrate unwanted influence. The same distinction applies to a near-match marker. Labeling either form negative is appropriate only for the stated exact event and should not be read as declaring the response harmless. Retaining both equality and substring endpoints makes that limitation visible without moving the success criterion after observing the response.

5.3. Implications for Reporting and Model Judges

A useful report separates structural validity, legitimate-answer utility, attacker-goal completion and evaluator decision coverage. Each quantity can deteriorate independently. A model that rarely outputs the marker can still fail the legitimate task. A judge that rarely returns a binary verdict can look accurate on its selected decisions while covering little of the dataset. Invalid output deserves a separate category rather than an automatic unsuccessful-attack label, especially when comparing procedures that parse structure with procedures that merely search raw text.

The judge conditions here offer two concrete forms of assistance: a trusted parser projection and a more explicit rubric. Their paired comparison is informative about the tested prompts, but it does not isolate a universal reasoning skill. In these calls, projection reduces correct Llama judgments from 23/48 to 15/48 but increases correct Gemma judgments from 36/48 to 48/48. The explicit rubric similarly changes these totals to 17/48 and 44/48, respectively. Neither intervention is a uniformly effective remedy across the tested packages. Projection removes evidence and supplies validity, while the explicit rubric supplies operational rules. The class distribution and individual errors remain more informative than a single headline agreement percentage. For an endpoint fully determined by application code, a model judge is

unnecessary as a primary reference; model judgments remain useful objects of audit or tools for outcomes that cannot be expressed by such a predicate.

A practical evaluation can therefore state the endpoint first, retain unedited outputs, test quoted and denied instructions, inspect escaped and duplicate keys, include no-instruction controls, and report failures to decide. These checks complement the task-relative evaluation practices in existing benchmarks [2–4, 9]. They should be adapted to the actual application rather than applied as a universal definition of injection success.

6. Limitations

The study has three local model packages, one generation per condition, short English source material and four fixed non-adaptive attack formulations. Quantization, runtime and community packaging limit comparability with other releases. No repeated-generation variance, multilingual generalization, adaptive attacker search, tool execution, exfiltration or production-system performance is estimated. Fixed seeds do not remove all implementation nondeterminism.

The public extension uses 80 distinct documents selected by length and balance criteria. Questions originally tied to individual documents can become ambiguous in the mixed retrieval corpus. The source-document hit rate is modest and is not answer-chunk recall. Normalized benchmark matching is an imperfect utility proxy, and low utility limits deployment relevance. The attacker controls a retrieved chunk after ranking; the experiment does not show that an attacker can cause that chunk to be retrieved.

The reference is exact-goal ground truth only within the stated contract. It does not establish semantic correctness or all forms of influence. Hexadecimal marker design contributes to short-string fuzzy matches. The quotation ablation has only 16 documents per package and does not guarantee instruction compliance. Evidence substring and length checks do not establish factual support. The judge sample is small, its positive class is not enriched after collection, and the judges also appear as target packages. There is no independent human annotation or external expert validation.

The extension was planned after the pilot, and its local hash freeze is not preregistration. Supplemental evidence-compliance checks, lexical decompositions and utility transitions were added during collection and are labeled exploratory. Reported intervals condition on the selected design and do not justify population-wide prevalence estimates or family-wide safety rankings.

7. Conclusion

This study audits the transfer of prompt-injection scoring procedures into an application with separate answer and evidence fields. The controlled pilot and public-corpus extension show why response representation and the matching predicate need to be specified together. Evidence quotation can create whole-response positives without exact answer overwrite, while answer projection can expose short-string partial matches or retain longer-answer mismatches. An executable endpoint, preserved outputs, no-instruction controls and explicit invalid/non-decision categories make these differences inspectable. Individual model judgments also respond differently to parser projection and explicit rules, requiring separate reporting of false positives and non-

decisions. The resulting measurements describe the tested contract and local packages; they are not a general injection defense or a claim that prior benchmarks used incorrect goals.

Data and Code Availability

The accompanying anonymous supplementary archive contains the pilot and extension protocols, frozen selections, licensed source-data excerpts, raw requests and returned responses, package metadata, analysis code, tests, source tables and editable figure exports. Data provenance and third-party licensing are included. No public repository deposit or permanent identifier is claimed.

Use of Generative AI

OpenAI Codex assisted with research planning, literature organization, software implementation, experiment execution, data analysis and manuscript drafting and revision. Experimental target and judge models are identified in the Methods. No independent human annotation or external peer review is represented by the automated checks. Author approval and the personal contribution, funding and competing-interest declarations are handled in the separate author-information document.

References

- [1] Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. *arXiv preprint arXiv:2302.12173*, version 2. <https://doi.org/10.48550/arXiv.2302.12173>
- [2] Yi, J., Xie, Y., Zhu, B., Kiciman, E., Sun, G., Xie, X., & Wu, F. (2025). Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1* (pp. 1809–1820). ACM. <https://doi.org/10.1145/3690624.3709179>
- [3] DeBenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., & Tramèr, F. (2024). AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. *Advances in Neural Information Processing Systems*, 37, Datasets and Benchmarks Track. <https://doi.org/10.52202/079017-2636>
- [4] Geng, R., Yin, C., Wang, Y., Chen, Y., & Jia, J. (2026). PIArena: A Platform for Prompt Injection Evaluation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 33170–33192). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2026.acl-long.1533>
- [5] Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., & Hendrycks, D. (2024). HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. In *Proceedings of the 41st International Conference on Machine Learning (Vol. 235, pp. 35181–35224)*. *Proceedings of Machine Learning Research*. <https://proceedings.mlr.press/v235/mazeika24a.html>
- [6] Gao, Y. (2026). How Reliable Is Your Jailbreak Judge? Calibration and Adversarial Robustness of Automated ASR Scoring. *arXiv preprint arXiv:2606.25487*, version 2. <https://doi.org/10.48550/arXiv.2606.25487>
- [7] Zhang, X. (2026). How Sensitive Are Safety Benchmarks to Judge Configuration Choices? *arXiv preprint arXiv:2604.24074*, version 1. <https://doi.org/10.48550/arXiv.2604.24074>
- [8] Reynolds, B. (2026). Adversarial Pragmatics for AI Safety Evaluation: A Diagnostic Framework and Seed Benchmark for Language-Mediated Control. *arXiv preprint arXiv:2607.01153*, version 3. <https://doi.org/10.48550/arXiv.2607.01153>
- [9] Zhan, Q., Liang, Z., Ying, Z., & Kang, D. (2024). InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024* (pp. 10471–10506). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.findings-acl.624>
- [10] Li, H., Liu, X., Zhang, N., & Xiao, C. (2025). PIGuard: Prompt Injection Guardrail via Mitigating Overdefense for Free. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 30420–30437). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2025.acl-long.1468>
- [11] Kholkar, G., & Ahuja, R. (2025). CAPTURE: Context-Aware Prompt Injection Testing and Robustness Enhancement. In *Proceedings of the First Workshop on LLM Security (LLMSEC)* (pp. 176–188). Association for Computational Linguistics. <https://aclanthology.org/2025.llmsec-1.13/>
- [12] Pasupat, P., & Liang, P. (2015). Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1470–1480). Association for Computational Linguistics. <https://doi.org/10.3115/v1/P15-1142>
- [13] Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389. <https://doi.org/10.1561/15000000019>