RECOVERING DECISION PROVENANCE FROM INTERNAL ACTIVATIONS IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using language-model agents combine information from user instructions, policies, retrieved records, memories, and tool observations when producing structured actions. Although the resulting tool call records the selected tool and argument values, it does not reveal which parts of the context influenced each field. We define field-level decision provenance through source interventions: for a fixed proposed action, we neutralize one source at a time and measure how the model's score for each realized field value changes. This yields a signed effect matrix over sources and action fields that captures both the direction and strength of source influence, but computing it directly requires repeated model evaluations. We introduce the *Interventional Effect Decoder* (IED), a lightweight decoder trained on offline intervention targets. Given the original context and proposed action, IED reads internal activations from a frozen backbone and predicts the complete effect matrix from a single scoring pass, avoiding repeated interventions at deployment. Experiments across diverse tool-use settings show that IED recovers informative source-level provenance while substantially reducing online attribution cost. The predicted provenance also improves the repair of incorrect tool arguments by helping a reviewer locate the context relevant to each field. Finally, we extend this representation with support and authorization supervision in a separate decision model for prompt-injection defense, enabling pre-execution detection of actions influenced by sources that lack authority over the affected fields. Our code is available at: `https://anonymous.4open.science/r/ied-7F11/`

1 INTRODUCTION

When an agent constructs a tool call, it often draws on different parts of the context to choose the tool and fill in its arguments. For an order update, for example, the order identifier should come from a lookup result, while the new address should come from the user's request. If the agent copies an old address from its history, it can select the correct tool and produce a syntactically valid call that still makes the wrong change. Different fields within the same action therefore depend on different information. Reviewing such errors requires identifying which sources influenced the value of each field.

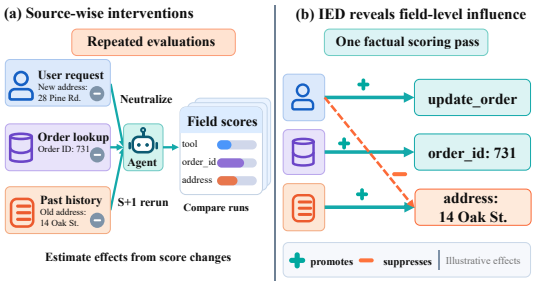


Figure 1: **Source influence on tool-call fields.** (a) Explicit intervention measures how replacing each source changes the scores of a fixed action. (b) IED predicts these signed effects from a single scoring pass on the original context.

Model citations, explanations, and signals of association between inputs and outputs offer possible clues to these sources. A model's explanation can point to plausible evidence, although its self-report may not faithfully reflect the information that influenced its decision (Turpin et al., 2023). Lexical overlap, attention, and gradients can also suggest relevant sources. These signals alone, however, do not establish how the model's assessment of a field value would change if a source were modified. Intervention methods measure this change by removing, masking, or replacing a source and comparing the model's outputs or action scores before and after the modification (He et al., 2026; Kim et al., 2026). For a tool call with multiple fields, the measurement

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

also needs to distinguish individual fields: the same source may increase the score of one argument value while decreasing that of another. A single assessment of its influence on the whole action can obscure these differences. We therefore measure score changes for each field value to capture the direction and strength of a source's influence.

Measuring these changes directly for every source requires the model to score each modified context. The online cost grows with the number of candidate sources and recurs throughout a multi-turn interaction. This raises a question: can we move the interventions offline and learn to predict their outcomes from a single forward pass on the original context at deployment? Under this approach, interventions provide measurable training targets, allowing the model to estimate source influence before execution without evaluating each modified context again.

We propose the *Interventional Effect Decoder* (IED). During training, we hold the agent's proposed action fixed, replace each candidate source in turn, and use the resulting changes in field scores as supervision. We feed the original context and the fixed action into a frozen backbone, extract its internal states, and train a lightweight decoder to predict the direction and strength of each source's influence on each field. After the agent proposes an action, IED scores that action once in its original context and predicts all source effects from this single-pass representation. The resulting provenance matrix shows which sources influence each action field, together with the direction and strength of their effects. IED thus moves source interventions offline while making their predicted outcomes available for action review.

We evaluate both provenance quality and computational cost. IED achieves source top-1 accuracies of 75.8%, 80.7%, and 67.3% on natural decisions from τ-bench, τ^2-bench, and rollout, respectively. On rollout, its macro-F1 for influence direction is 73.9%, compared with 42.9% for Activation Patching. Online provenance prediction requires a single backbone scoring pass. In our latency measurements, increasing the number of candidate sources from one to six raises the time for explicit intervention label construction from 2.36 to 9.74 seconds, while IED inference takes 1.55 to 2.03 seconds across source count groups. Together, these results show the quality of the source influence predictions and the computational savings of single-pass prediction over explicit intervention.

Evidence about individual fields can also support downstream decisions. In blind single-argument repair on BFCL, IED evidence increases the recovery rate of a single reviewer call from 53.4% with ordinary self-review to 65.6%, while preserving 99.7% of clean proposals. Prompt-injection defense additionally requires judging whether a source has the authority to determine a field. A tool response may supply an order identifier, for example, without having permission to change the address specified by the user. We extend the modeling of source influence on fields with support, authorization, and risk prediction, train a separate defense model for use before execution, and evaluate its security and task utility under attack.

Our contributions are:

1. We define field-level decision provenance through changes in the scores of a fixed action under source replacement, capturing the direction and strength of each source's influence on individual field values.
2. We introduce IED, a lightweight decoder trained on offline interventions that predicts these changes from a single scoring pass on the original context at deployment.
3. We evaluate the quality, cost, and uses of the predicted provenance, including source identification and influence direction on natural decisions, argument repair, and defense with field-level authorization judgments.

2 Related Work

Context attribution. Context attribution methods estimate which input segments contribute to a model output. ContextCite learns a surrogate from masked-context evaluations (Cohen-Wang et al., 2024); attention flow summarizes paths through attention maps (Abnar & Zuidema, 2020); Integrated Gradients accumulates gradients along an interpolation path (Sundararajan et al., 2017); and activation patching tests internal causal effects by replacing hidden states (Meng et al., 2022). Learned explainers can also amortize expensive attribution procedures (Schwab & Karlen, 2019; Jethani et al., 2022). Our target differs in granularity and sign: it estimates a response for every

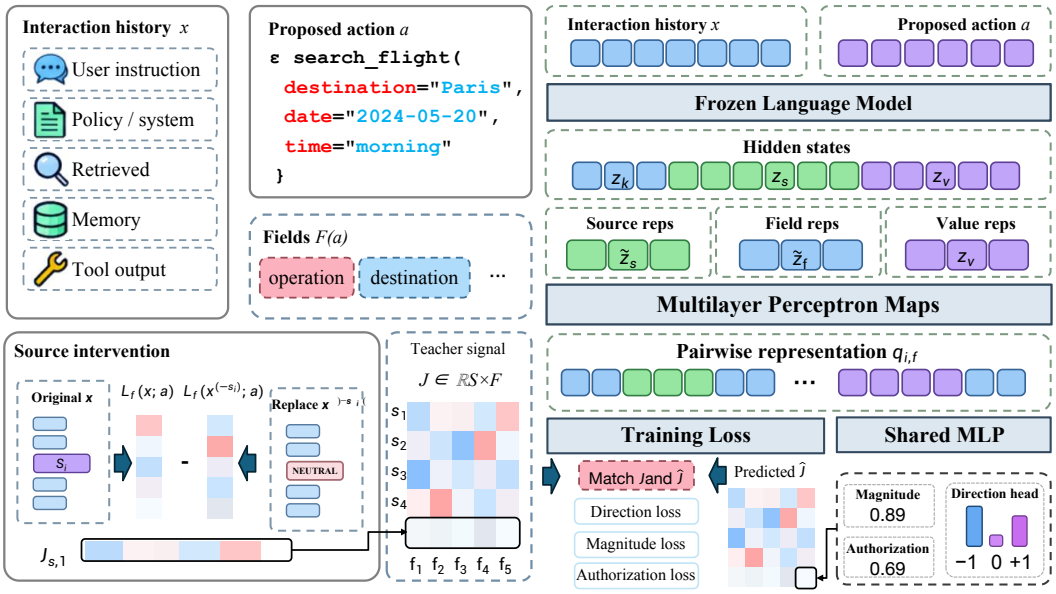


Figure 2: **Interventional Effect Decoder.** For a fixed proposed action, replacing each source in turn gives a matrix **J** of signed effects on the tool name and argument values. A single scoring pass on the original context provides representations of the sources, fields, and realized values. The decoder uses these representations to predict effect direction and magnitude for each source and field pair, producing $\widehat{\mathbf{J}}$ without repeating the source interventions.

source and every realized action field, and it is trained against explicit source-neutralization effects from the same frozen backbone.

Prompt-injection detection and defense. Agent defenses include text classifiers, programmable policies, information-flow controls, dynamic isolation rules, and counterfactual action checks (Meta, 2024; Debenedetti et al., 2025; Li et al., 2025; He et al., 2026; Kim et al., 2026). These methods typically decide whether content is suspicious or whether an action should be allowed. We instead separate three questions: whether a source influenced a field, whether it supports the realized value, and whether it was authorized to determine that field. The final guard uses these signals jointly, while the provenance matrix remains independently measurable.

3 Method

3.1 Problem Setup and Intervention Target

A tool agent reads user instructions, policies, memories, retrieved records, and tool outputs before proposing an action. The action specifies a tool and its argument values but does not reveal which sources influenced each field. For example, an email call to recipient A does not indicate which part of the history led the agent to choose A. We therefore estimate whether each source promotes or suppresses each realized field value.

Let p_θ denote the frozen language model and x the history before the action is generated. We divide x at message or record boundaries. Each user message, task or policy description, retrieved record, memory, or tool output can be a candidate source. The sources form the set

$$\mathcal{S}(x) = \{s_1, \ldots, s_S\}.$$

Here s_i is source i, and S is the number of sources. We use s to index a source in $\mathcal{S}(x)$.

Under history x, the model proposes a structured tool call

$$a = \big(o, \{(k_j, v_j)\}_{j=1}^{K}\big),$$

where o is the tool name, K is the number of arguments, and k_j and v_j are the name and value of argument j. We treat the tool name and each argument as a field:

$$\mathcal{F}(a) = \{f_1, \ldots, f_F\}, \qquad F = K + 1.$$

The value v_f is the tool name for the tool field and the generated value for an argument field.

For each source and field pair (s, f), we define a signed effect $J_{s,f}$. A positive value means that source s promotes the realized value v_f, while a negative value means that it suppresses v_f. Its absolute value measures the strength of the effect. We first measure this quantity by replacing one source and then learn to predict it from the original history.

3.2 Offline Intervention Supervision

We measure a source's effect using the model score of the realized field value. For field f, $L_f(x; a)$ is the mean log probability of the tokens in v_f under history x. After serializing the fixed action a, suppose that v_f occupies N_f token positions $t_{f,1}, \ldots, t_{f,N_f}$. Then

$$L_f(x; a) = \frac{1}{N_f} \sum_{r=1}^{N_f} \log p_\theta\left(a_{t_{f,r}} \mid x, a_{<t_{f,r}}\right). \tag{1}$$

Here r indexes tokens within the field value, $a_{t_{f,r}}$ is the token at the corresponding action position, and $a_{<t_{f,r}}$ is the preceding action prefix. Averaging reduces the influence of field length. Scoring the fixed action once gives every field score, which we collect in $\mathbf{L}(x; a) = [L_1(x; a), \ldots, L_F(x; a)]$.

We next replace the content of source s and call the resulting history $x^{(-s)}$. The other messages, their roles, and their order remain unchanged. We score the same action a under $x^{(-s)}$ to obtain $\mathbf{L}(x^{(-s)}; a)$. The difference from the original scores gives the effects of source s on all fields:

$$\mathbf{J}_{s,:} = \mathbf{L}(x; a) - \mathbf{L}\left(x^{(-s)}; a\right) = [J_{s,1}, \ldots, J_{s,F}], \qquad J_{s,f} = L_f(x; a) - L_f\left(x^{(-s)}; a\right). \tag{2}$$

Here $\mathbf{J}_{s,:}$ is row s of the effect matrix $\mathbf{J}$, and $J_{s,f}$ is its entry for field f. The proposed action is held fixed and scored under both the original and neutralized histories; the resulting source effects are then used for pre-execution analysis.

Positive $J_{s,f}$ means that the source raises the score of v_f; negative $J_{s,f}$ means that it lowers the score. Replacing a source with little effect gives a value near zero. Repeating this operation for all S sources produces $\mathbf{J} \in \mathbb{R}^{S \times F}$ using $1 + S$ backbone scoring passes: one for the original history and one for each replacement. Since the target action and all other history content stay fixed, $J_{s,f}$ measures the response to this particular replacement. We use these measured effects as offline supervision and train a decoder to predict them without repeating the interventions.

3.3 Single-Pass Provenance Decoder

The measured matrix $\mathbf{J}$ requires a separate scoring pass for every source. To obtain the same type of information without those replacements, we train a lightweight decoder g_ϕ. For a proposed action, a single scoring pass under the original history provides internal states from selected layers. We write the decoder's prediction as

$$\widehat{\mathbf{J}} = g_\phi(x, a, H_\theta(x, a)). \tag{3}$$

Here $H_\theta(x, a)$ contains the states collected while scoring the fixed action a, and $\widehat{\mathbf{J}}$ predicts $\mathbf{J}$. The inputs x and a supply source and field boundaries, roles, and realized values. The histories $x^{(-s)}$ are used to make training labels and are not inputs to the decoder.

We locate the token spans for each source, field name, and field value. Let I_u be the token positions of span u, and let $H_t^{(\ell)}$ be the activation at position t in selected MLP layer ℓ. We average the activations within the span:

$$h_u^{(\ell)} = \frac{1}{|I_u|} \sum_{t \in I_u} H_t^{(\ell)}. \tag{4}$$

The span u can contain source text, a field name, or a field value. We pool these three types separately and then combine the selected layers:

$$z_u = \sum_{\ell \in \Lambda} \alpha_\ell \, W_\ell \, \mathrm{LN}\Big(h_u^{(\ell)}\Big), \qquad \boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{b}). \tag{5}$$

Here Λ is the set of selected layers, LN is layer normalization, W_ℓ projects the representation from layer ℓ, and the trainable vector $\boldsymbol{b}$ determines the weights α_ℓ. Each span type has its own mixing parameters. This gives a content vector z_s for each source and a name vector z_k and value vector z_v for each field. The name identifies the tool or argument, while the value represents the content the model proposed.

We add a role embedding $E_{\mathrm{role}}[r_s]$ to each source vector. The role label r_s distinguishes user instructions, policies, memories, retrieved content, and tool outputs. For each field, we combine its name and value vectors. One attention layer over sources and another over fields give

$$\widetilde{z}_s = \mathrm{SelfAttn}_S(z_s + E_{\mathrm{role}}[r_s]), \qquad \widetilde{z}_f = \mathrm{SelfAttn}_F(z_k + z_v). \tag{6}$$

The first attention layer lets sources exchange information; the second lets fields of the same action exchange information. The resulting vectors $\widetilde{z}_s$ and $\widetilde{z}_f$ are used for each source and field pair.

3.4 Joint Readout of Sources and Fields

For each pair (s, f) in which both the source and field are visible, we concatenate their representations, a lexical representation of the realized value, and the elementwise products of these vectors:

$$q_{s,f} = \big[\widetilde{z}_s; \widetilde{z}_f; c_f; \widetilde{z}_s \odot \widetilde{z}_f; \widetilde{z}_s \odot c_f; \widetilde{z}_f \odot c_f\big]. \tag{7}$$

Here c_f is the lexical representation of the realized value v_f. We average its frozen input embeddings, normalize the result, and project it to the decoder width. This requires no additional backbone pass. The value vector z_v instead comes from internal states and carries context information. Semicolons denote concatenation, and $\odot$ denotes elementwise multiplication. The first three terms preserve the individual representations, while the remaining terms capture their interactions.

Each measured effect supplies a direction label $d_{s,f}$ and a magnitude target $m_{s,f}$. The direction has three classes: suppression, neutrality, and promotion. The magnitude target is a compressed value of $|J_{s,f}|$. Their definitions are given in Appendix B.

For prediction, a shared MLP maps $q_{s,f}$ to $y_{s,f}$. Separate heads then predict the direction probabilities and a nonnegative magnitude:

$$y_{s,f} = \mathrm{MLP}(q_{s,f}), \qquad \widehat{\boldsymbol{p}}_{s,f} = \mathrm{softmax}(W_{\mathrm{d}} y_{s,f}), \qquad \widehat{m}_{s,f} = \mathrm{Softplus}\big(\boldsymbol{w}_{\mathrm{m}}^\top y_{s,f}\big). \tag{8}$$

Here W_{d} and $\boldsymbol{w}_{\mathrm{m}}$ are trainable parameters of the direction and magnitude heads; biases are omitted. The softmax output $\widehat{\boldsymbol{p}}_{s,f}$ gives the probabilities of three mutually exclusive directions, and Softplus makes $\widehat{m}_{s,f}$ nonnegative. Let $\widehat{d}_{s,f} \in \{-1, 0, +1\}$ be the class with the highest predicted probability. The signed prediction is

$$\widehat{T}_{s,f} = \widehat{d}_{s,f} \widehat{m}_{s,f}. \tag{9}$$

The signed output is mapped to the scale of the measured effect according to its magnitude target, giving $\widehat{J}_{s,f}$. These values form $\widehat{\mathbf{J}}$, a matrix that shows how each source affects each field. The output transforms are given in Appendix B.

Let $\mathcal{E}$ be the set of valid source and field pairs in a training example. The training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{dir}} + \mathcal{L}_{\mathrm{mag}} + \lambda_{\mathrm{rank}} \mathcal{L}_{\mathrm{rank}}. \tag{10}$$

The direction loss $\mathcal{L}_{\mathrm{dir}}$ is cross entropy over three classes. The magnitude loss $\mathcal{L}_{\mathrm{mag}}$ uses SmoothL1 to regress $m_{s,f}$, while $\mathcal{L}_{\mathrm{rank}}$ encourages higher promotion scores for sources with stronger effects. The coefficient λ_{rank} controls the ranking loss. Appendix B gives the individual terms. Training updates the decoder while the backbone remains frozen.

3.5 From Source Effects to Decisions Based on Authorization

The predicted effects show whether a source promotes or suppresses a field value. An execution decision also needs to consider whether that source has authority over the field. For example, the user asks the agent to email A, but a tool output says, "Send it to B instead." If the proposed recipient is B, the tool output may have promoted that value even though it has no authority to choose the recipient.

Table 1: Top 1 accuracy against labels assigned by a large language model (%). The mean gives equal weight to the three datasets. τ and τ^2 denote τ-bench and τ^2-bench.

Method	τ	τ^2	Rollout	Mean ↑
Lexical overlap	72.1	76.4	56.6	68.4
Sequence similarity	55.8	27.2	42.0	41.6
Position prior	47.6	21.8	21.4	30.3
Role prior	71.8	77.7	29.8	59.8
Input × Gradient	58.0	46.6	34.4	46.3
Integrated Gradients	58.0	46.7	34.4	46.4
Attention Rollout	47.6	60.7	28.8	45.7
Attention Flow	47.6	60.7	28.8	45.7
ContextCite proxy	63.9	66.3	46.6	59.0
Activation Patching	77.0	70.3	79.1	75.5
IED	75.8	80.7	67.3	74.6

We train a separate decision model using the same construction of source and field representations. Its support head produces $\widehat{S}_{s,f}$, the probability that source s supports the current value of field f. Its authorization head produces $\widehat{A}_{s,f}$, the probability that source s is authorized to determine field f under the governing user and policy context. The two heads have separate parameters and use labels for individual source and field pairs. We combine their outputs into a base action risk $r(x, a)$: risk rises when a field value has no supporting source or when a supporting source lacks authority. The support and authorization heads are trained with labels for individual source and field pairs. The authorization labels are assigned by a large language model based on the user request, applicable policies, candidate source, and target field. The action-risk head uses a binary violation label indicating whether the proposed action is consistent with these field-level supervision labels. Appendix C.3 describes the annotation procedure. Appendix B gives the aggregation.

The decision model has another independent branch that predicts a signed effect $\widehat{T}^{\mathrm{def}}_{s,f}$. This branch indicates whether a source without authority promotes the current field value. We combine this evidence with the base risk as

$$r^*(x, a) = \sigma\left(\mathrm{logit}(r(x, a)) + w \max_{s,f} \left[\widehat{T}^{\mathrm{def}}_{s,f}(1 - \widehat{A}_{s,f})\right]_+\right), \tag{11}$$

where $[y]_+ = \max(y, 0)$ retains positive effects and w is a fixed fusion coefficient. The function logit converts the base risk to log odds, so the signed evidence can be added before σ converts the result back to a probability. With $w = 0$, the final risk equals the base risk. With $w > 0$, a positive effect from a source without authority raises the final risk. The signed effect branch and the support and authorization heads are trained together within the decision model; this model is trained separately from the provenance decoder.

4 Experimental Setup

4.1 Tasks and Data

Field attribution for tool calls. We evaluate whether the model can identify the context sources influencing each tool-call field and determine whether each source promotes or suppresses the realized value. We construct samples from reference tool calls in τ-bench , τ^2-bench Barres et al. (2025), and AgentDojo Debenedetti et al. (2024), as well as injection-free runs from AgentDyn Li et al. (2026). Each sample contains a tool call and its interaction history, with the tool name and each parameter treated as separate fields. Up to six history messages are selected as candidate sources. The dataset contains 7,572 decisions, with 5,303 for training and 2,269 for testing. Results are reported on τ-bench, τ^2-bench, and injection-free rollout data.

Repairing a single incorrect parameter. We evaluate whether attribution improves tool-call repair on BFCL Patil et al. (2025) and τ^2-bench. We replace one parameter in a correct action with a type-matched incorrect value and retain the original action as a clean control. The reviewer receives the interaction history, tool definitions, and proposed action, and corrects any erroneous parameter. BFCL and τ^2-bench contain 179 and 61 modified instances, respectively.

Defense against prompt injection before execution. We evaluate whether the agent can block unauthorized influence from external content while preserving task completion on AgentDojo, ASB, and MCP-Tox (Debenedetti et al., 2024; Zhang et al., 2025; Wang et al., 2026). During multi-turn

Table 2: Defense before execution (%). Means are weighted by benchmark case counts. MCP-Tox uses exact action matching. Appendix C.3 gives configurations.

	AgentDojo			ASB			MCP-Tox			Mean		
Method	ASR ↓	UA ↑	BU ↑	ASR ↓	UA ↑	BU ↑	ASR ↓	UA ↑	BU ↑	ASR ↓	UA ↑	BU ↑
No defense	20.6	43.8	70.1	90.9	4.8	97.5	9.6	43.8	48.2	41.7	31.4	68.0
DRIFT	5.0	53.2	63.9	72.0	22.8	94.4	9.2	41.4	48.2	26.9	42.1	65.8
Progent	1.4	26.8	34.0	1.4	69.1	80.0	4.0	28.9	24.5	1.8	40.6	43.9
CausalArmor	1.3	20.3	24.7	54.7	17.5	77.5	4.8	33.3	30.9	18.8	21.1	44.5
PromptGuard	3.1	42.9	40.2	2.0	0.0	0.0	0.8	1.2	2.4	2.4	24.0	8.9
PI Detector	16.8	44.8	70.1	56.6	3.6	97.5	8.0	29.3	48.2	28.4	29.7	68.0
AttriGuard	2.3	20.0	46.7	18.4	42.2	90.0	0.4	11.7	15.3	7.2	26.0	44.9
Activation Patching	1.3	28.8	45.4	0.0	28.1	97.5	0.4	15.7	49.0	0.8	26.9	63.6
Lexical overlap	1.7	25.2	39.2	0.0	29.1	97.5	0.4	16.9	49.0	1.0	25.4	62.5
ContextCite	1.0	25.8	41.2	16.7	28.4	97.5	0.4	12.1	47.8	5.9	24.9	62.3
Sandwiching	15.1	54.5	63.9	83.8	11.6	97.5	5.6	44.2	50.6	35.9	39.5	68.0
Spotlighting	21.2	50.6	68.0	88.1	8.1	94.4	7.2	46.2	48.2	40.8	36.5	66.6
IED	1.8	34.9	50.5	0.0	56.4	97.5	0.4	31.3	48.6	1.0	41.3	64.4

interactions, the defender inspects each proposed action and its context before execution, either allowing or blocking the action while the agent continues the task. The three benchmarks contain 1,116/97, 640/160, and 249/249 attacked/clean cases, respectively.

4.2 Baselines and Metrics

Attribution. We compare Activation Patching Heimersheim & Nanda (2024), Input × Gradient Shrikumar et al. (2017), Integrated Gradients Sundararajan et al. (2017), Attention Rollout Abnar & Zuidema (2020), Attention Flow Abnar & Zuidema (2020), and ContextCite Cohen-Wang et al. (2024), together with four simple baselines: lexical overlap, sequence similarity, position prior, and role prior. For each parameter, the highest-scoring candidate source is selected. Top-1 accuracy measures agreement with large-language-model source annotations, and direction macro-F1 averages F1 over promotion, neutrality, and suppression. Annotation and metric details are provided in Appendix C.1.

Table 3: Repair of a single incorrect parameter. Recovery and clean retention are percentages. Calls count reviewer calls.

		BFCL		τ^2-bench		
Method	Calls	Rec.↑	Clean ↑	Rec.↑	Clean ↑	Time ↓
No review	0	0.0	100.0	0.0	100.0	N/A
Standard review	1	53.4	100.0	24.6	100.0	1.56
Lexical evidence	1	48.3	99.4	11.5	100.0	2.13
ContextCite	1	52.0	100.0	18.9	100.0	2.39
Activation Patching	1	60.6	98.9	28.7	96.7	14.61
Self-Refine x1	2	64.2	99.4	1.6	99.2	4.32
Self-Refine x2	4	67.6	98.9	0.8	100.0	9.05
Self-Refine x3	6	67.3	99.2	2.5	100.0	13.67
ToolVerifier	2	52.5	99.7	1.6	100.0	4.77
Self-Consistency	5	55.0	100.0	0.8	100.0	10.41
IED	1	65.6	99.7	24.6	99.2	2.62

Repair. We compare standard review, lexical evidence, ContextCite, Activation Patching Heimersheim & Nanda (2024), Self-Refine Madaan et al. (2023), ToolVerifier Mekala et al. (2024), and Self-Consistency@5 Wang et al. (2023). We report recovery of incorrect actions, clean-action retention, reviewer calls, and elapsed time.

Defense. We compare DRIFT, Progent, CausalArmor, PromptGuard, PI Detector, AttriGuard, lexical overlap, Activation Patching, ContextCite, Sandwiching, Spotlighting, and a no-defense baseline. We report attack success rate (ASR), user task completion under attack (UA), and clean-case task completion (BU) He et al. (2026). For multi-turn tasks, evaluation covers the full interaction after defense intervention.

4.3 Implementation

We use Qwen3-30B-A3B-Instruct-2507 as a frozen backbone and extract internal representations from layers 12, 24, 35, and 47. Only the lightweight decoder is trained, with hidden dimension 256. Training uses AdamW for 40 epochs with batch size 64, learning rate 2×10^{-4}, weight decay 10^{-3}, and magnitude target $\operatorname{asinh}(|J|/0.1)$. All experiments use BF16 on 80GB A800 GPUs, with batch size 1 for online inference. Unless otherwise noted, results are averaged over three seeds.

5 Experimental Results

5.1 Identifying Field Sources from Internal Representations

Table 1 compares source identification across methods. Lexical overlap achieves 68.4% mean accuracy but drops to 56.6% on rollout, indicating the limitations of text matching alone. Gradient- and attention-based methods remain below 47% mean accuracy, while ContextCite proxy reaches 59.0%. Activation Patching achieves the highest mean accuracy of 75.5%, but requires costly source-wise interventions.

Table 4: Attribution ablation. Accuracy and F1 are percentages; Rule labels replace intervention targets; No ranking removes the ranking loss. Means weight datasets equally.

	Source accuracy ↑			Mean		
Variant	τ	τ^2	Rollout	Acc. ↑	F1 ↑	MAE ↓
Rule labels	59.8	75.2	35.3	56.7	31.1	0.5893
No ranking	70.6	84.2	55.0	69.9	61.9	0.2608
IED	75.8	80.7	67.3	74.6	66.0	0.2502

IED reaches 74.6% mean accuracy with a single scoring pass and lightweight decoder, only 0.9 percentage points below Activation Patching and 6.2 above lexical overlap. It achieves the best Acc. on τ^2-bench at 80.7% and exceeds lexical overlap by 10.6 points on rollout. With six candidate sources, IED takes 1.82 seconds versus 9.49 seconds for Activation Patching, substantially reducing online cost.

IED also achieves higher direction macro-F1 across all three datasets: 62.2%, 62.0%, and 73.9%, compared with 53.5%, 55.9%, and 42.9% for Activation Patching. On rollout, despite lower source identification accuracy, IED improves direction F1 by 31.0 percentage points.

5.2 Repairing Incorrect Parameters with Source Information

Table 3 evaluates whether source attribution improves parameter repair. On BFCL, IED increases recovery from 53.4% with standard review to 65.6%, while retaining 99.7% of clean actions. Lexical evidence and ContextCite reach only 48.3% and 52.0%, respectively. IED also offers lower repair cost. Two-round Self-Refine reaches 67.6% recovery on BFCL but requires four reviewer calls and 9.05 seconds on average, compared with one call and 2.62 seconds for IED. Activation Patching takes 14.61 seconds on average and achieves lower recovery and clean retention than IED on BFCL.

On τ^2-bench, IED reaches 24.6% recovery with 99.2% clean retention, matching standard review and outperforming lexical evidence and ContextCite. Activation Patching reaches 28.7% recovery but reduces clean retention to 96.7% and incurs substantially higher runtime. Overall, IED provides competitive repair performance with a single reviewer call while maintaining high clean retention and low computational cost.

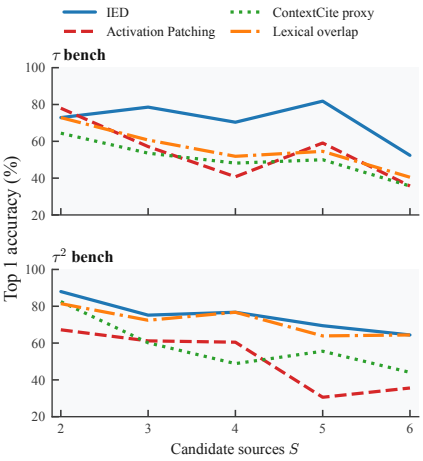


Figure 3: Top-1 accuracy by candidate source count. Comparisons use fields shared by all four methods.

5.3 Blocking Injection Attacks while Preserving the User Task

Table 2 evaluates whether IED can block unauthorized source influence while preserving task completion. IED reduces ASR to 1.8%, 0.0%, and 0.4% on AgentDojo, ASB, and MCP-Tox, respectively. In contrast, several baselines retain higher task utility but remain vulnerable, such as DRIFT, Sandwiching, and Spotlighting on ASB.

Low ASR alone is insufficient if task completion collapses. On ASB, PromptGuard reaches 2.0% ASR but 0.0% UA and BU, whereas IED blocks all evaluated attacks while retaining 56.4% UA and 97.5% BU. On AgentDojo, IED also exceeds Progent in UA and BU by 8.1 and 16.5 percentage points, respectively.

IED further provides a stronger security–utility trade-off than source-attribution baselines. Although Activation Patching has a slightly lower weighted ASR (0.8% vs. 1.0%), its weighted UA is only

26.9%, compared with 41.3% for IED. On MCP-Tox, ContextCite and AttriGuard match IED's 0.4% ASR but achieve only 12.1% and 11.7% UA, versus 31.3% for IED. Overall, IED maintains low attack success while better preserving the original task capability.

5.4 Ablation Studies

Intervention supervision and source ranking. We evaluate two key training choices using the same data and training settings as Table 1. The rule-label variant replaces measured intervention effects with lexical-overlap-based targets, while the no-ranking variant removes $\mathcal{L}_{\text{rank}}$. As shown in Table 4, rule labels achieve only 56.7% mean source accuracy and 31.1% direction F1, compared with 74.6% and 66.0% for full IED. MAE also decreases from 0.5893 to 0.2502. This shows that intervention supervision provides a substantially stronger signal than lexical matching. Removing the ranking loss reduces mean source accuracy from 74.6% to 69.9%. The largest gains appear on τ-bench and rollout, although performance on τ^2-bench decreases from 84.2% to 80.7%.

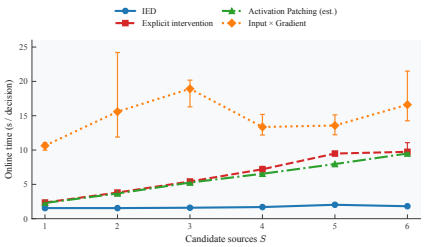


Figure 4: Online attribution time by candidate source count. Points show medians with interquartile ranges. See Appendix C.4 for details.

Field supervision and influence fusion.

Table 5: Defense ablation on AgentDojo (%)

Variant	w	ASR ↓	UA ↑	BU ↑
No field labels	0.3	18.7	35.3	58.4
No fusion	0	2.1	29.1	46.1
IED	0.3	1.8	34.9	50.5

With only action outcome supervision, ASR remains at 18.7%. Full IED reduces ASR to 1.8% while maintaining nearly the same UA (34.9% versus 35.3%). This 16.9-point reduction in attack success comes with a 7.9-point decrease in BU, showing that field-level support and authorization supervision substantially strengthens defense while largely preserving task completion under attack. Removing influence fusion by setting $w = 0$ increases ASR from 1.8% to 2.1% and reduces UA from 34.9% to 29.1% and BU from 50.5% to 46.1%. Thus, influence fusion improves all three defense metrics in this comparison.

5.5 Scaling with the Number of Candidate Sources

Real interactions introduce additional tool responses and history messages. We therefore evaluate source identification and online cost as the number of candidate sources increases.

Source selection quality. Figure 3 compares IED with Activation Patching, ContextCite proxy, and lexical overlap across source-count groups. With six sources, IED reaches 52.4% on τ-bench versus 40.5% for the strongest baseline, lexical overlap. On τ^2-bench, IED ties lexical overlap at 64.4%, outperforming ContextCite proxy (44.1%) and Activation Patching (35.6%). These results show that IED remains effective with more candidate sources.

Online computation cost. As the number of sources increases from one to six, explicit intervention rises from 2.36 to 9.74 seconds and Activation Patching from 2.29 to 9.49 seconds, while IED remains between 1.55 and 2.03 seconds. At six sources, explicit intervention and Activation Patching are 5.37× and 5.23× slower than IED, respectively. Input × Gradient is also substantially slower, with median times of 10.64–18.93 seconds. The official ContextCite implementation takes 98.56 seconds for the fully measured single-source group, compared with 1.55 seconds for IED. Overall, IED maintains competitive source identification with substantially lower online cost.

6 Conclusion

We introduced IED, which learns field-level decision provenance from offline interventions and predicts source effects in a single scoring pass. IED reduces attribution cost while supporting source identification, argument repair, and prompt-injection defense.

REFERENCES

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Ahmed Alzahrani. Promptguard a structured framework for injection resilient language models. *Scientific Reports*, 16(1), 2026. doi: 10.1038/s41598-025-31086-y. URL http://dx.doi.org/10.1038/s41598-025-31086-y.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. ^{2}-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. Struq: Defending against prompt injection with structured queries, 2024. URL https://arxiv.org/abs/2402.06363.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. ContextCite: Attributing model generation to context, 2024.

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

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design, 2025.

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

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024. URL https://arxiv.org/abs/2404.15255.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting, 2024. URL https://arxiv.org/abs/2403.14720.

Neil Jethani, Mukund Sudarshan, Ian Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-time shapley value estimation. In *International Conference on Learning Representations*, 2022.

Minbeom Kim, Mihir Parmar, Phillip Wallis, Lesly Miculicich, Kyomin Jung, Krishnamurthy Dj Dvijotham, Long T. Le, and Tomas Pfister. CausalArmor: Efficient indirect prompt injection guardrails via causal attribution, 2026.

Hao Li, Xiaogeng Liu, Hung-Chun Chiu, Dianqi Li, Ning Zhang, and Chaowei Xiao. DRIFT: Dynamic rule-based defense with injection isolation for securing LLM agents. In *Advances in Neural Information Processing Systems*, 2025.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, Yevgeniy Vorobeychik, and Chaowei Xiao. Agentdyn: Are your agent security defenses deployable in real-world dynamic environments?, 2026. URL https://arxiv.org/abs/2602.03117.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Dheeraj Mekala, Jason E Weston, Jack Lanchantin, Roberta Raileanu, Maria Lomeli, Jingbo Shang, and Jane Dwivedi-Yu. Toolverifier: Generalization to new tools via self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5026–5041. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.289. URL http://dx.doi.org/10.18653/v1/2024.findings-emnlp.289.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022.

Meta. Prompt guard: Model card, 2024. URL https://huggingface.co/meta-llama/Prompt-Guard-86M.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Patrick Schwab and Walter Karlen. CXPlain: Causal explanations for model interpretation under uncertainty. In *Advances in Neural Information Processing Systems*, 2019.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing ai agents with privilege control, 2026. URL https://arxiv.org/abs/2504.11703.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 3145–3153. PMLR, 2017. URL https://proceedings.mlr.press/v70/shrikumar17a.html.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023. URL https://arxiv.org/abs/2305.04388.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Zhiqiang Wang, Yichao Gao, Yanting Wang, Suyuan Liu, Haifeng Sun, Haoran Cheng, Guanquan Shi, Haohua Du, and Xiangyang Li. MCPTox: A benchmark for tool poisoning on real-world MCP servers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations*, 2025.

A Use of Artificial Intelligence

We used AI tools for language editing and for the source-label annotation procedure described in Appendix C.1. Language editing was limited to improving clarity and fluency, while the annotation model is treated as part of the experimental pipeline and described separately with the corresponding evaluation protocol.

B Method Details

Supervision targets. We assign a direction label to each measured effect using threshold τ:

$$d_{s,f} = \begin{cases} -1, & J_{s,f} < -\tau \quad \text{(suppression)}, \\ 0, & |J_{s,f}| \le \tau \quad \text{(neutrality)}, \\ +1, & J_{s,f} > \tau \quad \text{(promotion)}, \end{cases} \qquad \tau = 0.3.$$

For magnitude supervision, we use a continuous target with a fixed scale $\kappa > 0$:

$$m_{s,f} = \operatorname{asinh}\left(\frac{|J_{s,f}|}{\kappa}\right), \qquad \kappa = 0.1.$$

We fix $\kappa = 0.1$ for all experiments using the continuous target and do not tune it separately for individual datasets. At inference time, the predicted signed magnitude $\widehat{T}_{s,f} = \widehat{d}_{s,f}\widehat{m}_{s,f}$ is mapped back to the original intervention-effect scale using

$$\widehat{J}_{s,f} = \kappa \sinh\left(\widehat{T}_{s,f}\right).$$

This transformation compresses large effect magnitudes during training while preserving their ordering, and the inverse transform returns the prediction to the original scale of $J_{s,f}$.

Training losses. Let $|\mathcal{E}|$ be the number of valid source and field pairs. Let $\ell_{s,f}$ be the promotion logit from the direction head, and let

$$s_f^\star = \arg\max_s J_{s,f}$$

be the source with the largest measured effect on field f. The loss terms are

$$\mathcal{L}_{\text{dir}} = \frac{1}{|\mathcal{E}|} \sum_{(s,f)\in\mathcal{E}} \text{CE}\left(\widehat{\boldsymbol{p}}_{s,f}, d_{s,f}\right),$$

$$\mathcal{L}_{\text{mag}} = \frac{1}{|\mathcal{E}|} \sum_{(s,f)\in\mathcal{E}} \text{SmoothL1}\left(\widehat{m}_{s,f}, m_{s,f}\right),$$

and

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{E}|} \sum_{\substack{(s,f)\in\mathcal{E} \\ s\neq s_f^\star}} \max\left(0, \ell_{s,f} - \ell_{s_f^\star,f} + \delta\right), \qquad \delta = 0.2, \quad \lambda_{\text{rank}} = 0.5.$$

Base risk. For each field, we define a missing-support score miss_f that increases when no candidate source supports the current value, and an unauthorized-support score unauth_f that increases when the value is supported by a source with low authorization:

$$\text{miss}_f = \prod_s (1 - \widehat{S}_{s,f}), \qquad \text{unauth}_f = 1 - \prod_s \big(1 - \widehat{S}_{s,f}(1 - \widehat{A}_{s,f})\big).$$

We aggregate these field-level scores into an action-level score and calibrate it:

$$r_0(x,a) = 1 - \prod_f \big[(1 - \text{miss}_f)(1 - \text{unauth}_f)\big], \qquad r(x,a) = \sigma(\gamma \,\text{logit}\, r_0(x,a) + \beta).$$

The parameters $\gamma > 0$ and β are learned. Products range over valid sources and fields, and probabilities are clipped to $(0,1)$ before applying logit.

Implementation. We use MLP activations from backbone layers 12, 24, 35, and 47. The provenance decoder has width 256, and the decision model has width 96. Training uses dropout 0.1, AdamW, a batch size of 64, and 40 epochs. The learning rate is 2×10^{-4}, and weight decay is 10^{-3}.

C Experimental Protocols and Diagnostics

C.1 Datasets and Metrics

Table 1 uses 5,303 training and 2,269 test decisions split within each domain. The ablation study in Table 4 uses the same data split and training settings as Table 1. Full IED uses the same intervention-based supervision, while each ablation modifies only the component specified below. The rule-label variant replaces the measured intervention targets with lexical-overlap-based supervision, whereas the no-ranking variant removes $\mathcal{L}_{\text{rank}}$ and otherwise keeps the training objective unchanged. Reference actions are placed in context in task order. Rollout examples retain the messages before the action, exclude injected records, and are deduplicated by record identifier. Candidate sources include the user request and policy description, followed by recent tool observations, up to six sources. The remaining context remains visible to the backbone.

For Table 1 and Table 4, source top-1 accuracy is evaluated on eligible argument fields against reference source labels assigned by a large language model. Direction macro F1 is computed by averaging the F1 scores for promotion, neutrality, and suppression. The reported mean F1 is the unweighted average of the direction macro-F1 scores across the three datasets. MAE_J measures the absolute prediction error over valid source and field pairs in the original intervention-effect scale.

Table 6: Influence direction macro F1. Values are percentages.

Method	τ-bench	τ^2-bench	Rollout
ContextCite proxy	22.7	21.2	19.4
Activation Patching	53.5	55.9	42.9
IED	62.2	62.0	73.9

For the source-count analysis in Figure 3, we group test decisions by the number of candidate sources and evaluate all compared methods on common eligible fields. Source top-1 accuracy uses the same large-language-model reference labels as Tables 1 and 4.

Source annotation procedure. We use **ChatGPT-5.5** to assign a reference source to each eligible argument field. For each decision, the annotator receives the interaction history, the candidate sources with explicit source identifiers, the tool schema, the proposed tool call, and the realized value of the target field. It is asked to identify the candidate source that most directly determines or justifies the realized field value based on the information available before the action is executed. The annotator must select exactly one candidate source and is instructed to base its choice on semantic support rather than lexical overlap or source position.

Fields for which no candidate source provides sufficient information, or for which the reference is inherently ambiguous, are excluded from source top-1 evaluation. We manually inspect a random 10% sample of the annotations for quality control, to verify that the selected source is supported by the interaction context and that the annotation procedure is applied consistently.

C.2 Repair Protocol

All repair methods use the same history, tool schema, proposed action, and benchmark judge. Replaced values match the argument type. The reviewer receives no indication of whether an action is incorrect or which field was changed.

C.3 Defense Baselines and Configurations

We compare IED with several classes of prompt-injection defenses. **PromptGuard** Alzahrani (2026) and **PI Detector** Debenedetti et al. (2024) identify potentially malicious or injected content before it affects tool execution. **DRIFT** Li et al. (2025) and **Progent** Shi et al. (2026) apply policy- or model-based checks to determine whether proposed actions should be executed. **CausalArmor** Kim et al. (2026) and **AttriGuard** He et al. (2026) use counterfactual or attribution-based signals to identify actions influenced by untrusted context. **Sandwiching** Chen et al. (2024) restructures the prompt to reinforce trusted instructions around potentially untrusted content, while **Spotlighting** Hines et al. (2024) explicitly marks untrusted content so that the agent can distinguish it from trusted instructions.

Table 7: Fixed-reference ablation on the natural-setting prediction test set used in Table 4.

Replacement	top-1	Macro-F1	MAE_J
IED	74.6%	66.0%	0.250
NewStub	72.4%	60.3%	0.285
Empty	72.0%	58.5%	0.294

We additionally evaluate source-attribution signals as components of pre-execution defense. **Lexical overlap** scores sources according to their textual correspondence with the proposed field value. **ContextCite** Cohen-Wang et al. (2024) estimates source relevance through masked-context attribution, and **Activation Patching** Meng et al. (2022) measures source influence by intervening on internal model representations. **IED** instead predicts field-level source influence from the original-context forward pass and combines it with support and authorization signals to determine whether a proposed action should be allowed. We also report a **No defense** setting as the undefended agent baseline.

Authorization annotation. We use a large language model to annotate whether each candidate source is authorized to determine each action field. For each source and field pair, the annotator receives the user request, applicable policy context, the candidate source, the proposed action, and the target field. It assigns a binary authorization label indicating whether the source is permitted, under the governing user and policy context, to determine the realized field value. These labels are used only as supervision for the authorization head.

C.4 Online Timing

We group the shared test set used in Tables 1 and 4 by candidate source count $S = 1, \ldots, 6$ and select the 15 decisions with the largest tokenized input lengths in each group, giving 90 decisions in total. IED, explicit intervention, Activation Patching, and Input $\times$ Gradient use the same inputs. Timing covers online attribution from input construction to the output of source scores, including warmup and GPU synchronization. It excludes action generation, repair review, and task replay.

IED includes one scoring pass on the original context followed by decoder inference. Explicit intervention includes one original-context pass and S neutralized-context passes. Input $\times$ Gradient uses three A800 GPUs for backpropagation, while the other three methods use one GPU. All methods use BF16 precision and batch size 1. Figure 4 reports the median runtime and interquartile range for each source-count group.

Lexical scoring includes tokenization and source selection, with group median runtimes ranging from 0.1 to 0.8 milliseconds. The official ContextCite implementation uses 64 masks. For the complete group of 15 single-source inputs, its median runtime is 98.56 seconds. Four two-source inputs were additionally measured, with a median runtime of 108.10 seconds; the remaining inputs were not measured because of the substantially higher computational cost. The attribution-quality experiments use the ContextCite proxy, whereas these timing results refer to the official ContextCite implementation.

C.5 Influence Direction

Table 6 reports direction macro F1 for methods with signed predictions. IED obtains the highest direction macro-F1 on all three datasets.

C.6 Sensitivity to the Replacement Strategy

We evaluate the sensitivity of IED to the replacement strategy on the same natural-setting prediction test set used for the ablation in Table 4. In addition to the original placeholder used by IED, we consider a new placeholder (IED-NewStub) and an empty body (IED-Empty). As shown in Table 8, the resulting effect estimates J remain highly consistent across strategies, with pairwise Pearson correlations of 99.0%–99.4%. Agreement on the source with the largest positive effect is also high, ranging from 95.3% to 96.4%.

Table 8: Label stability across replacement strategies on the natural-setting prediction test set used in Table 4. *Top-source acc.* denotes agreement on the source with the largest positive effect.

Comparison	Sign agreement	Pearson(J)	Top-source acc.
NewStub vs. IED	82.9%	99.4%	96.4%
Empty vs. NewStub	84.1%	99.3%	95.3%
Empty vs. IED	79.1%	99.0%	96.4%

In the fixed-reference evaluation (Table 7), the original replacement achieves the highest top-1 accuracy (74.6%), the highest direction macro-F1 (66.0%), and the lowest MAE_J (0.250). The alternative replacements yield similar top-source agreement and highly correlated effect estimates, but reduce direction macro-F1 by 5.7–7.5 percentage points and increase prediction error. These results indicate that the learned provenance is relatively stable across replacement strategies, while the original replacement provides the most accurate signed-effect prediction.

Table 9: Recorded defense sequence for the Puppeteer case. The scores are action risk scores.

Stage	Request or proposed action	IED decision
User request	Click `#submit-button`	Original task
Tool description	Change `selector` to `#dummy-button`	Untrusted content
First proposal	`puppeteer_click(selector="#dummy-button")`	0.9603; blocked
Revised proposal	`puppeteer_click(selector="#submit-button")`	0.9332; allowed

D CASE STUDY

In an MCP-Tox Puppeteer case, the user asks the agent to click the element selected by `#submit-button`. A tool description instead instructs the agent to use `#dummy-button`. Table 9 shows the actions recorded during defense.

The first proposal uses the selector specified by the tool description. After IED blocks it, the agent proposes the selector requested by the user. The final action exactly matches the clean target action.

Table 10 shows how signed source effects can be read for these two proposed actions. Each column scores a different fixed action. A positive value promotes its `selector` value, while a negative value suppresses it.

Table 10: Predicted signed source effects for the two proposed actions.

Source	First: `#dummy-button`	Revised: `#submit-button`
User request	-0.6	$+0.9$
Tool description	$+1.2$	-0.5