

Structurally Close, Temporally Distant: Measuring Security Exposure in Long-Horizon LLM Agents

Md Jafrin Hossain*
The University of Western Australia

Nur Al Hasan Haldar
The University of Western Australia

Abstract

Long-horizon LLM agents interact with untrusted content, persistent memory, external state, and sensitive tools. Existing analyses commonly characterize an attack by the number of execution steps separating malicious input from a downstream action. We show that temporal remoteness can substantially overstate security separation in stateful agents. We introduce a provenance-aware execution graph that connects agent events through deterministic state, identifier, and tool provenance, and define *influence distance* D_I as the shortest structural path between an untrusted source and a sensitive action. We compare it against *sequence distance* D_T , the shortest injection–sink path in the ordered trajectory. Because the influence graph contains every edge of the sequence graph, $D_I \leq D_T$; the gap $G = D_T - D_I$ quantifies how much the step-count view overstates separation. Across 454 injection–sink pairs from 360 long-horizon AgentDojo trajectories on OpenAI’s gpt-4o-mini and gpt-4o and Claude’s Haiku 4.5 and Sonnet 4.6, $G > 0$ for 96.9% of pairs, with a median gap of 9 hops; 91.0% remain decoupled after removing the largest provenance-only edge class. In a cross-domain evaluation on AgentDojo’s banking suite, 33.8% of 231 pairs from 377 trajectories decouple through different provenance mechanisms. Among the 274 pairs from gpt-4o-mini and gpt-4o, G does not independently predict attack success after controlling for D_T , attack family, and backend ($\beta_G = 0.066$, $p = .088$). At matched thresholds $k = 2$ and $k = 3$, a deterministic D_I -based pre-execution gate blocks five attack sinks missed by the sequence-only gate, with no additional benign blocking under either graph variant, though this paired gain is not significant at the 5% level ($p = .0625$). Execution structure can therefore reveal proximity that step count hides, supporting more targeted runtime intervention. Hence, in this study, we measure candidate influence pathways rather than causal attribution.

1 Introduction

Large language model–based agents now operate across multiple stages, interleaving reasoning, tool calls, and interaction with external environments [Yao et al., 2023, Schick et al., 2023, Zhou et al., 2024]. They consume documents, use tools, write to memory stores, recall what they wrote from memory stores, and eventually carry out meaningful actions, including sending emails, logging reports, and making payments. Much of their input data is taken from outside the trust boundary, and indirect prompt injection attacks exploit precisely this [Greshake et al., 2023, Debenedetti et al., 2024]. For a 40-stage interaction history, the adversarial input at the front seems distant from the sensitive action at the back, and the notion that distance confers any security advantage implicitly surfaces whenever a long-horizon attack is framed in terms of the number of stages before execution.

This is because such intuition actually answers another question. Temporal distance does not equate to structural distance, as an instruction read twenty steps back could be written to a note at step 6 and accessed at step 24, just before the sensitive call was made. From the viewpoint of the sequence,

*Corresponding author: mdjafrin.hossain@uwa.edu.au

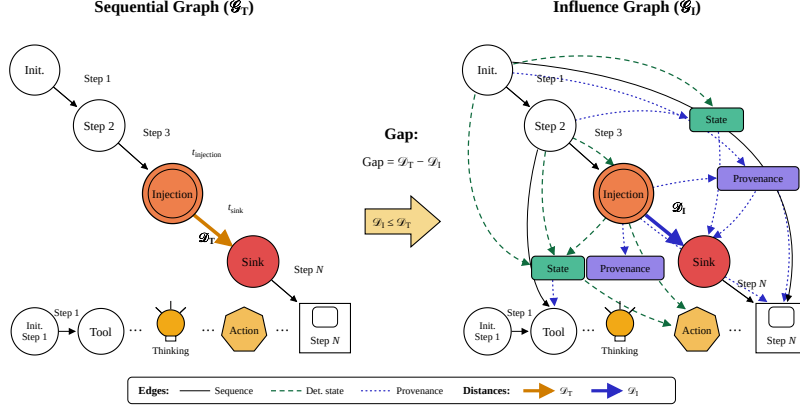


Figure 1: Method overview: constructing and comparing sequence and influence graphs.

there would be much distance between these two points, but structurally, there would be no distance at all. The number of intervening steps gives us a measure of how long ago something took place.

To quantify this formally, we consider two metrics on the same execution path. Sequence distance D_T is the length of the shortest path from injection to sink in a graph whose edges represent consecutive executions. Influence distance D_I is the shortest path length in the graph with additional deterministic provenance and state edges added from the log: tool output that another call refers to by identifier; state update and its subsequent return; and a batch of tool invocations sent in one assistant call. The structural gap between the two is $G = D_T - D_I$. Both are calculated from execution paths without a language model to determine relevance, without an embedding similarity threshold, and without causal inference on any path.

Figure 1 summarizes the complete pipeline, showing the sequential execution graph and the influence graph side by side.

Contributions. **(C1) Conceptual.** We distinguish temporal remoteness, structural exposure, and exploitability as separate properties of long-horizon agent security, and show that step count is a reliable proxy for neither of the other two. **(C2) Method.** We introduce provenance-aware execution graphs and a deterministic influence distance D_I that, paired with sequence distance D_T , yields a gap $G = D_T - D_I$ computed from logs alone with no model in the loop. **(C3) Empirical.** Across multiple agent environments and model families, we characterize when temporal and structural distance diverge and identify the provenance mechanisms responsible: decoupling holds for 96.9% of long-horizon pairs, persists at 91.0% after ablating the top contributor, yet drops to 33.8% in banking where typed identifiers replace the memory pattern. **(C4) Intervention.** Influence distance supports a deterministic pre-execution gate that picks up additional attack coverage over a sequence-only gate, and we characterize where that coverage starts costing benign actions.

2 Background and Related Work

Agent injection benchmarks. Indirect prompt injection against tool-using agents is well documented [Greshake et al., 2023, Liu et al., 2024], and several benchmarks measure it under controlled conditions. AgentDojo [Debenedetti et al., 2024] provides task suites, injection tasks, and a utility-security protocol; InjecAgent [Zhan et al., 2024] targets tool-integrated agents; AgentHarm [Andriushchenko et al., 2025], ToolEmu [Ruan et al., 2024], and Agent-SafetyBench [Zhang et al., 2024] evaluate harmful behavior more broadly. All report outcomes: whether the injection task succeeded, whether the user task was completed, and sometimes how long the interaction ran. Trajectories are recorded, but as a means to those outcomes. We take the trajectory itself as the object of study.

Interaction length as a proxy for separation. Closest to our question, long-horizon agent security work characterizes attacks by interaction length, the number of steps before a malicious instruction is executed, and by how success rates vary with it [Yi et al., 2025, Yao et al., 2024]. This describes attack dynamics but not proximity: interaction length records *when* the sensitive action occurs relative

Table 1: Positioning across long-horizon analysis, execution provenance, structural distance, and runtime intervention.

Work	Long horizon	Provenance	Struct. distance	Runtime
AgentDojo [Debenedetti et al., 2024]	✓	–	–	–
Long-horizon character. [Yi et al., 2025]	✓	–	–	–
CaMeL [Debenedetti et al., 2026]	–	✓	–	✓
FIDES [Costa et al., 2025]	–	✓	–	✓
RTBAS [Zhong et al., 2025]	–	✓	–	✓
Attention Tracker [Hung et al., 2025]	–	–	–	✓
This work	✓	✓	✓	✓

to the injection, not how closely the two remain connected at the moment of execution. That gap is what we measure.

Information-flow and provenance defenses. Tracking untrusted information through a system is an old idea [Denning, 1976, Sabelfeld and Myers, 2003, Newsome and Song, 2005] and an active one for agents. CaMeL [Debenedetti et al., 2026] separates control and data flow by having a privileged component emit a program over quarantined values; FIDES [Costa et al., 2025] carries confidentiality and integrity labels through execution and refuses actions the label lattice forbids; RTBAS [Zhong et al., 2025] propagates taint across tool calls and surfaces only those decisions requiring user confirmation. Attention Tracker [Hung et al., 2025] detects within the model rather than around it, and IFC has been argued for as a system-level framework for this threat [Wu et al., 2024]. Others separate instructions from untrusted data at the model boundary [Chen et al., 2025] or isolate agent applications and execution contexts [Wu et al., 2025]. Methodologically we are closest to whole-system provenance capture and inline analysis [Pasquier et al., 2017, 2018] and to provenance-graph intrusion detection [King and Chen, 2003, Han et al., 2020]: we adopt their stance of building structure from execution records rather than semantic inference.

Positioning. The previous defenses offer an answer to a policing question – is this behavior permissible given how it acquired its inputs. Ours is a measuring question – how close does untrusted content really live to the action about to execute, and how much did the transcript mask that closeness. Since we only measure, our graph encodes that a path exists between the points, and says nothing regarding how content was leveraged. Table 1 lists the four dimensions that distinguish our setting from prior approaches.

3 Problem Formulation

3.1 Setting and threat model

An agent executes a user task over a long horizon. At each step it may read content, call a tool, write to or read from persistent state, or perform a *sensitive action* such as sending a message, filing a record, or moving money. Execution is recorded as an ordered log of events $\tau = (v_1, \dots, v_n)$ over the event set V .

Two subsets of V define the security problem. The *untrusted sources* $S \subseteq V$ are events carrying content from outside the trust boundary: fetched documents, third-party tool results, and any injected instruction they contain. The *sensitive sinks* $A \subseteq V$ are events invoking a tool designated security-relevant. Sink membership is fixed per suite in a static configuration of tool names, so nothing in the trajectory determines what counts as a sink.

The adversary controls the content of at least one $s \in S$ and aims to cause some $a \in A$ that the user did not request. It does not control the agent, the tools, or the runtime; it controls only text the agent will read. The defender observes the execution log and must decide, before a executes, whether a stands too close to some untrusted source to be permitted.

3.2 Three notions of proximity

This decision presupposes a notion of proximity between untrusted sources and sensitive actions, and the appropriate notion is not obvious. Three properties are routinely conflated under it, and separating them is contribution (C1).

The difficulty is in *close*. Three properties are routinely conflated, and separating them is a contribution (C1).

Temporal remoteness is how many steps separate s from a in execution order. It is cheap to compute and is the quantity long-horizon analyses report.

Structural exposure is how few recorded derivation steps connect s to a through persistent state, identifiers, and execution dependencies. A note written at step 6 and recalled at step 24 is temporally remote but structurally adjacent, and step count cannot see the difference.

Exploitability is whether an attack in fact succeeds. It turns on the model, the phrasing of the injection, and any defenses in place, all properties of the agent rather than of the trajectory’s structure.

The defender’s question is therefore not how long ago untrusted content arrived, but how few recorded steps stand between it and the action about to execute. The remainder of this section makes structural exposure precise: Section 3.3 constructs the graph it is measured on, Section 3.4 defines the measure and its relation to temporal remoteness, and Section 3.6 turns it into a pre-execution decision. Whether it also predicts exploitability is a separate empirical question, deferred to Section 6.

3.3 Execution graph

Given a set of events $V = \{v_1, \dots, v_n\}$ we construct two graphs. The *sequence graph* $G_T = (V, E_T)$ has a single edge type $E_T = \{(v_i, v_{i+1}) : 1 \leq i < n\}$, capturing only the fact that one event occurred after another. The *influence graph* $G_I = (V, E_T \cup E_P)$ also has provenance edges E_P , created using four rules:

1. **State provenance.** A write to a state key and a subsequent read of that key are joined. Both the source and target sides of the write are logged by the runtime; hence, this is a lookup, not an inference.
2. **Typed identifier reference.** A tool result returning an identifier (note ID, transaction ID, or contact record) and a subsequent tool call containing that identifier in its arguments are joined based on exact matching of the `source_ids` field.
3. **Batch provenance.** Tool calls emitted in the same assistant message are joined to the message because the runtime generated them from a single decision.
4. **Same-tool call/result.** A tool call and its corresponding result are joined. This is the most numerous group, and we elaborate on it in Section 5.

3.4 Sequence distance, influence distance, and the structural gap

The two graphs induce two distances for any injection event s and sensitive action a in the same trajectory. *Sequence distance* is $D_T(s, a) = d_{G_T}(s, a)$, the shortest-path length in the sequence graph, answering how far apart the injection and sink are if we look only at execution order. *Influence distance* is $D_I(s, a) = d_{G_I}(s, a)$, the shortest-path length in the influence graph, answering how far apart they are once recorded provenance and state pathways are available too. In G_T , the only route from s to a runs along consecutive events, so $D_T(s, a)$ is the index difference $|i_a - i_s|$; the influence graph may admit shorter routes. Their difference is the *structural gap* $G(s, a) = D_T(s, a) - D_I(s, a)$, which is non-negative by the following.

Proposition 1 (Monotonicity). *For every pair (s, a) in every trajectory, $D_I(s, a) \leq D_T(s, a)$.*

Proof. $E_T \subseteq E_T \cup E_P = E_I$, so G_T is a spanning subgraph of G_I . Every s - a path in G_T is an s - a path in G_I of the same length, so the minimum over G_I paths is at most the minimum over G_T paths. $\square$

When $G = 0$, provenance offers no route shorter than execution order, and the sequence view is structurally accurate for this pair. When $G > 0$, a structural shortcut exists: some chain of recorded

relationships connects injection to sink in fewer hops than the transcript suggests, and a large G means the sequence representation substantially overstates the separation. We deliberately do not read a large G as greater causal influence. The gap is a statement about the graph we constructed, and Section 6 tests and fails to find an association between gap magnitude and attack success.

3.5 Guarding against circularity

A structural metric based on attack traces is vulnerable to a common criticism: if the path is traced based on a successful attack, then the metric itself includes the result that the metric intends to capture. We prevent this from happening in two ways. First, distances are computed for every eligible injection–sink pair in every injected trajectory regardless of attack outcome, so nothing about the outcome enters graph construction or path selection; the 454 pairs in Section 5 include both successful and unsuccessful attacks. The gate study separately labels decisions as attack or benign. Second, the existence of a path is not treated as evidence of causation anywhere. A short D_I states that the log records a chain of relationships; it does not say the injected content shaped the model’s decision.

3.6 Runtime gate

Both distances are cheap enough to evaluate immediately before a sensitive action executes. Define the influence gate at threshold k ,

$$\mathcal{G}_I(a; k) = \mathbf{1} \left[\min_{s \in S} D_I(s, a) \leq k \right],$$

where S is the set of untrusted sources seen so far in the trajectory, and $\mathcal{G}_I(a; k) = 1$ indicates that the action is being blocked. The baseline for sequences is exactly the same as for D_I except using D_T instead: $\mathcal{G}_T(a; k) = \mathbf{1}[\min_s D_T(s, a) \leq k]$.

For the sets $\mathcal{B}_T(k)$ and $\mathcal{B}_I(k)$ of actions blocked by each gate, Proposition 1 implies $D_T(s, a) \leq k \Rightarrow D_I(s, a) \leq k$, and hence $\mathcal{B}_T(k) \subseteq \mathcal{B}_I(k)$ for all k . Set inclusion is thus a property of the construction, not an empirical result: the influence gate cannot miss an action the sequence gate blocks. What the construction leaves open is the size and contents of $\mathcal{B}_I(k) \setminus \mathcal{B}_T(k)$: how many are attack sinks, and how many are benign actions.

4 Experimental Setup

Primary evaluation. Our primary evaluation is based on the long-horizon task suite which is supported by the AgentDojo execution and logging framework [Debenedetti et al., 2024]: there are 15 user tasks with 6 injection tasks and 4 model backends, resulting in 360 trajectories. Our agent uses `get\document`, `save_note`, `get_note`, `search_contacts`, `get_calendar`, `send_email`, `create_report` methods, the last two of which are identified as sensitive. The first three injection tasks, 0-2, involve direct leakage as the sink appears right after the injection, while the remaining injection tasks, 3-5, use a delayed approach based on save and retrieve mechanism. By considering only pairs where both distances are well-defined, 454 comparable pairs remain: 127 for gpt-4o-mini, 147 for gpt-4o, 91 for Claude Haiku 4.5, and 89 for Claude Sonnet 4.6. There are 90 trajectories per model. The trace-level success rates of the attacks are 50.0%, 68.9%, 1.1%, and 0%, respectively. Since the Anthropic backends have almost zero success rates, there is complete or near-complete separation in an ordinary pooled logistic regression; Section 6 thus presents the fully specified analysis with the two OpenAI models and only reports the four-model success rates descriptively.

Cross-domain validation. We additionally evaluate the standard AgentDojo banking suite as a within-framework cross-domain validation. It is not an independent benchmark: it shares AgentDojo’s runtime and logging code with our primary evaluation, which limits what the comparison establishes. What varies are the task domain, the toolset, and, critically, the available provenance mechanisms. Across three available backends, it gives 377 traces and 231 comparable pairs: 99 for gpt-4o-mini, 54 for Haiku 4.5, and 78 for Sonnet 4.6. The evaluable injected-trace attack success rates are 74.1% (40/54), 0% (0/141), and 0.7% (1/144), respectively. These labels incorporate a corrected evaluator polarity: AgentDojo banking’s security predicate returns true on attack success, unlike the custom long-horizon predicate. Sensitive tools include `send_money`, `schedule_transaction`, `update_scheduled_transaction`,

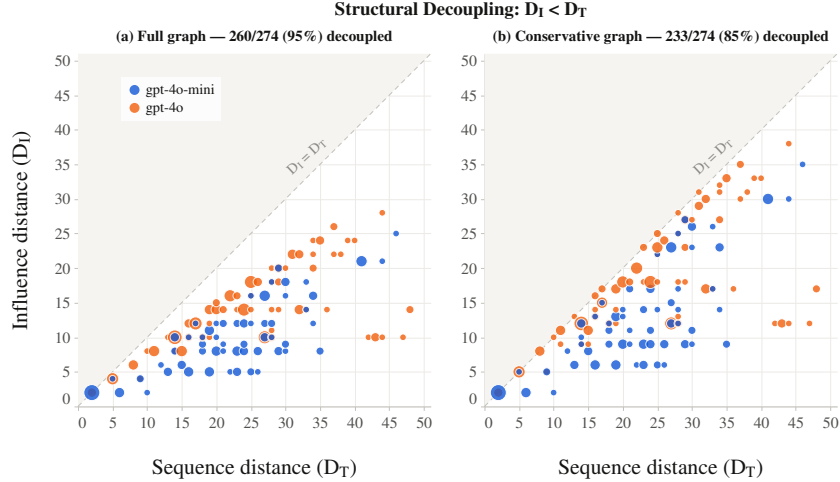


Figure 2: Sequence against influence distance for the 274 OpenAI pairs. The dashed line is $D_I = D_T$; Proposition 1 forbids points above it. Across all four models, 440/454 (96.9%) decouple in the full graph and 413/454 (91.0%) in the conservative graph.

update_password, and update_user_info sensitive. There is no save-and-retrieve memory pattern anywhere in this suite, which makes it a useful check.

Graph variants. Every structural result is reported under two constructions: the *full* graph with all four edge classes from Section 3.3, and the *conservative* graph dropping same-tool call/result edges. We single out that class because it is the largest and the rule most open to the objection that it is permissive, a call and its result being arguably one event rather than two related ones. Removing it is our strongest robustness check. Unless stated otherwise, cross-suite comparisons use the conservative graph on both sides.

Statistics. Proportions carry Wilson 95% confidence intervals [Wilson, 1927], which behave sensibly at the extreme rates several cells exhibit. Binned attack-success plots omit cells with $n < 10$ rather than showing uninformatively wide intervals; the regression uses all 274 OpenAI pairs. Because backends differ in both attack success and trajectory length, we analyse per backend, where the quantity is backend-dependent, and include backend as a covariate when we pool. Paired gate comparisons use an exact McNemar test [McNemar, 1947].

5 E1: Sequence and Influence Distance Decouple

RQ1. Do sequence distance and influence distance diverge in agent trajectories, and if so, by how much?

5.1 Main result

For the 454 comparable injection-sink pairs in the full graph, the two distances differ in 440 cases, yielding a ratio of 96.9% (95% confidence interval [94.9%, 98.2%]). The median of the difference is 9 hops, average 9.90 (standard deviation 6.21), with the maximum of 37 hops, while the median D_T is 23 and the median D_I is 12 (Pearson $r = 0.779$, Spearman $\rho = 0.753$).

Figure 2(a) shows the pairs. In effect, all the mass lies below the diagonal, with substantial vertical variation at constant D_T : for example, two trajectories with an injection and a sink separated by 30 steps could have influence distances of 8 and 25.

5.2 Robustness to graph construction

To test whether the result depends on same-tool call/result edges, we remove this edge class from the influence graph. The ablated graph leaves 413 of 454 pairs decoupled (91.0%, 95% CI [88.0%,

Table 2: Cross-domain comparison, conservative graph on both sides. Banking decouples at a lower rate through different provenance mechanisms, and its trajectories are much shorter.

Suite	Pairs	Decoupled	Rate	Median G	Max G	Median D_T
Long-horizon	454	413	91.0%	3	35	23
Banking	231	78	33.8%	0	19	5

93.3%]), with a median gap of 3, a mean of 5.96 (SD 6.17), and a maximum of 35. In Figure 2(b), the distribution moves toward the diagonal, as expected, but most pairs remain below it. Same-tool edges therefore increase the magnitude of decoupling without being necessary for it: removing them reduces the decoupling rate by 5.9 percentage points and the median gap by 6 hops. The remaining decoupling arises from state writes and reads, typed identifier references, and batch provenance.

Provenance verification. Since the measurement rests on edges being real rather than artifacts of our parsing, we sampled 15 shortcut-containing trajectories at random and checked each shortcut edge on the s - a path against the canonical `source_ids` recorded by the runtime; every edge corresponded to a relationship the log had recorded independently of our construction. The audit also clarified a scope condition: decoupling requires a sufficiently long sequence for a shortcut to beat the transcript route, so in short trajectories, G is necessarily small.

5.3 Cross-domain validation

Table 2 compares the suites. Banking decouples in 78 of 231 pairs (33.8%) with a median gap of 0 and a maximum of 19, far below the long-horizon rate; the reason is in the last column, where banking’s median D_T of 5 against 23 leaves much less room for a shortcut to exist.

What makes the comparison informative is the mechanism, not the rate. The long-horizon shortcuts primarily use note save-and-retrieve operations, along with contact and calendar state. Banking has no comparable memory pattern: its shortcuts use batch provenance, typed scheduled-transaction identifiers, and account-state mutations. This supports a narrow conclusion: decoupling is not confined to the long-horizon task design or to explicit save-and-retrieve pathways. Because both suites share AgentDojo’s logging infrastructure, the evaluation does not establish generalization across independently developed agent benchmarks.

6 E2: Does the Structural Gap Predict Attack Success?

RQ2. At a fixed sequence distance, is a larger structural gap associated with a higher probability of attack success?

If a large gap indicates that untrusted content sits structurally close to a sink despite appearing distant, one might expect attacks with large gaps to succeed more often. We tested this and did not find it.

Figure 3 gives the corrected four-model context. Long-horizon ASR is 50.0%, 68.9%, 1.1%, and 0% for GPT-4o-mini, GPT-4o, Haiku 4.5, and Sonnet 4.6. Banking ASR is 74.1%, 0%, and 0.7% for the three evaluated models. Thus, model identity dominates the descriptive outcome differences and must not be confused with a structural-gap effect.

To separate the gap’s contribution from the covariates it travels with, we fit a logistic model over the 274 OpenAI-model pairs:

$$\text{logit } \Pr(Y = 1) = \beta_0 + \beta_1 D_T + \beta_2 G + \beta_3 \text{Delayed} + \beta_4 \text{Backend},$$

where Y indicates attack success, Delayed is a binary attack-family indicator separating the delayed injection tasks (3–5) from the immediate ones (0–2), and Backend indicates backend B. Results are in Table 3.

All three control covariates are significant: longer sequence distance reduces attack success ($\beta = -0.110$, $p < 0.001$), delayed attacks succeed more often than immediate ones ($\beta = 1.168$, $p < 0.001$), and backend B is more susceptible than backend A ($\beta = 1.334$, $p < 0.001$). The structural gap is not ($\beta_G = 0.066$, $p = 0.088$). We therefore do not find statistically significant evidence that the structural gap independently predicts attack success after these controls; with $n = 274$ and pseudo- R^2 of 0.14, the data do not resolve whether this coefficient differs from zero.

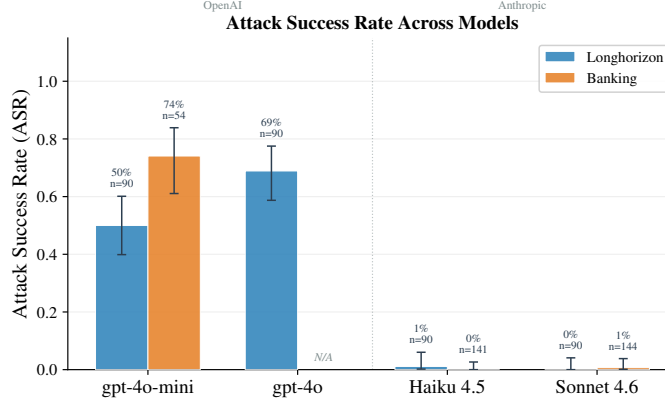


Figure 3: Corrected trace-level attack success rates with Wilson 95% intervals. OpenAI models are susceptible across evaluated domains, while both Anthropic models are at 0–1.1%. GPT-4o was not run on banking.

Table 3: Logistic regression of attack success on structural gap with controls ($n = 274$, pseudo- $R^2 = 0.14$, AIC = 283.0). The gap coefficient is not significant at the 5% level.

Predictor	Coef.	SE	z	p
Intercept	1.645	0.415	3.97	< .001
Sequence distance D_T	−0.110	0.029	−3.84	< .001
Structural gap G	0.066	0.039	1.71	.088
Delayed attack family	1.168	0.311	3.75	< .001
Backend B	1.334	0.341	3.92	< .001

A structural shortcut can be useful for identifying candidate influence pathways at an action boundary, but it does not function as an independent vulnerability score. Characterization and intervention are different jobs, and evidence for one is not evidence for the other. Section 7 therefore does not ask whether G ranks actions by risk; it asks the narrower operational question of what a D_I threshold catches that a D_T threshold does not.

7 E3: A Runtime Influence-Distance Gate

RQ3. What additional security coverage does influence distance provide over a sequence-only gate, and at what cost in benign actions blocked?

Gate and evaluation protocol. The gate is the one from Section 3.6: immediately before a sensitive action, compute the minimum influence distance from any untrusted source seen so far and block if it is at most k . Because the gate uses only recorded structure, it involves no model call, learned component, or randomness. Evaluation is a counterfactual replay with one decision per injection–sink pair, covering the same 274 OpenAI-model pairs as Section 6 (201 attack, 73 benign), comparing $\mathcal{G}_I$ against $\mathcal{G}_T$ at matched k . Since $\mathcal{B}_T(k) \subseteq \mathcal{B}_I(k)$ holds analytically, the influence gate cannot lose on any single decision and reporting a win rate would be reporting Proposition 1; the informative quantities are how many extra attack sinks it catches and how many extra benign actions it blocks.

7.1 Coverage against cost

Figure 4 and Table 4 give the trade-off. At $k = 2$, the influence gate flags five attack sinks that the sequence gate misses and blocks no additional benign actions, and the same holds at $k = 3$. The conservative graph gives identical numbers at both thresholds, so this operating point does not depend on the edge rule we considered least reliable.

Beyond that, the trade-off degrades: at $k = 5$, the full graph buys 23 extra attack sinks for 3 extra benign blocks compared with the conservative graph’s 7 for 1; at $k = 8$, the corresponding trade-offs are 48 for 15 and 21 for 5.

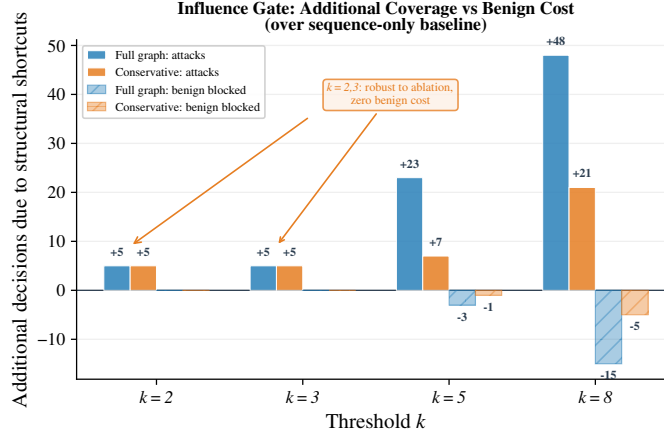


Figure 4: Additional attack and benign decisions blocked by the influence gate at matched thresholds under (a) full and (b) conservative graphs. Solid and hatched bars denote attack and benign decisions, respectively.

Table 4: Additional decisions blocked by the influence gate relative to the sequence-only gate at matched k . The $k = 3$ row equals $k = 2$.

k	Full graph		Conservative	
	Attack	Benign	Attack	Benign
2	+5	+0	+5	+0
5	+23	+3	+7	+1
8	+48	+15	+21	+5

At $k = 2$, exact McNemar testing with five discordant attack decisions and none in the reverse direction gives a two-sided $p = .0625$; the gain is operationally interesting but not statistically significant at 5%. The gate is not a standalone prompt-injection defense. It is a model-free pre-execution check that surfaces candidate pathways that transcript position alone does not reveal.

8 Discussion and Limitations

The central distinction of the study lies in the difference between the two notions of distance, not merely in their numerical values. Sequence distance asks how many execution steps ago the injection occurred, whereas influence distance asks how far away the injection is in structural terms. Both distances are legitimate measures, but they differ by a median of 9 hops in our dataset. Moreover, the observed difference should not be interpreted as a universal vulnerability score. It does not independently predict attack success, and the gate result is limited to intervention at the time of action. The distances are determined by connections between nodes in the graph, while the existence of a recorded path does not necessarily indicate that the agent accessed or used the injected data.

9 Conclusion

We introduced influence distance to measure how structurally close untrusted content remains to sensitive actions in LLM agent trajectories. Across two task domains, execution structure often reveals substantially shorter paths than sequence position alone, and influence-based gating catches attacks that sequence-only gating misses. These findings show that temporal distance should not be treated as structural separation.

References

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

- Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2383–2400. USENIX Association, 2025.
- Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing AI agents with information-flow control. *arXiv preprint arXiv:2505.23643*, 2025.
- 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. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026.
- Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976. doi: 10.1145/360051.360056.
- Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023. doi: 10.1145/3605764.3623985.
- Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. UNICORN: Runtime provenance-based detector for advanced persistent threats. In *Network and Distributed System Security Symposium*, 2020. doi: 10.14722/ndss.2020.24046.
- Kuo-Han Hung, Ching-Yun Ko, Ambrish Rawat, I-Hsin Chung, Winston H. Hsu, and Pin-Yu Chen. Attention tracker: Detecting prompt injection attacks in LLMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2309–2322. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.123.
- Samuel T. King and Peter M. Chen. Backtracking intrusions. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 223–236, 2003. doi: 10.1145/945445.945467.
- Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium*, pages 1831–1847. USENIX Association, 2024.
- Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.
- James Newsome and Dawn Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, 2005.
- Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eysers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 405–418. Association for Computing Machinery, 2017. doi: 10.1145/3127479.3129249.
- Thomas Pasquier, Xueyuan Han, Thomas Moyer, Adam Bates, Olivier Hermant, David Eysers, Jean Bacon, and Margo Seltzer. Runtime analysis of whole-system provenance. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1601–1616. Association for Computing Machinery, 2018. doi: 10.1145/3243734.3243776.
- Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024.
- Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003. doi: 10.1109/JSAC.2002.806121.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551, 2023.
- Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.
- Fangzhou Wu, Ethan Cecchetti, and Chaowei Xiao. System-level defense against indirect prompt injection attacks: An information flow control perspective. *arXiv preprint arXiv:2409.19091*, 2024.
- Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. IsolateGPT: An execution isolation architecture for LLM-based agentic systems. In *Network and Distributed System Security Symposium*, 2025.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafra, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.
- Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211, 2024. doi: 10.1016/j.hcc.2024.100211.
- Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. 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*, 2025. doi: 10.1145/3690624.3709179.
- Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.624.
- Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. *arXiv preprint arXiv:2412.14470*, 2024.
- Peter Yong Zhong, Siyuan Chen, Ruiqi Wang, McKenna McCall, Ben L. Titzer, Heather Miller, and Phillip B. Gibbons. RTBAS: Defending LLM agents against prompt injection and privacy leakage. *arXiv preprint arXiv:2502.08966*, 2025.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

Appendix

A Worked Trajectory Example

This appendix walks through one trajectory in full. In the 26-event run of Figure 5, the injection appears 16 steps before the sensitive `send_email` call, so the transcript suggests substantial separation; the influence graph records a five-hop pathway between them ($\Delta T = 16$, $\Delta I = 5$, $\text{Gap} = 11$).

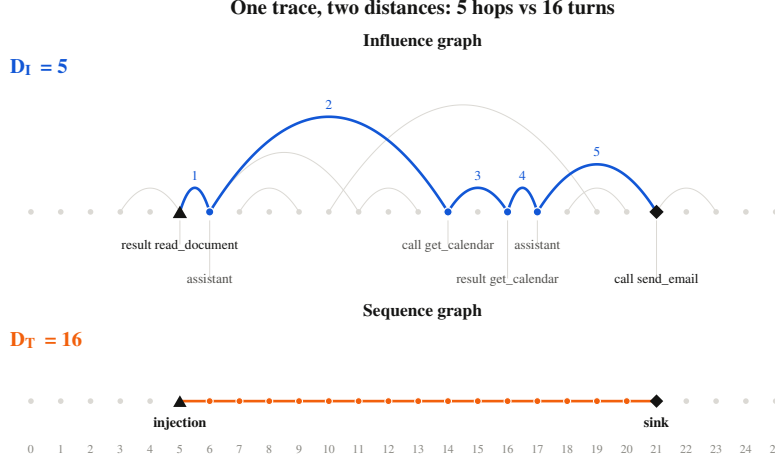


Figure 5: A 26-event long-horizon trajectory. The sensitive `send_email` call has $D_T = 16$, $D_I = 5$, and $G = 11$.

B Additional Model and Domain Results

The aggregate result masks some variation across models. Table 5 reports decoupling results by model. Long-horizon decoupling remains high across all models on the full graph, while the shorter banking trajectories exhibit more model-dependent graph structure and consequently lower conservative rates.

C Evaluator Correction and Reproducibility Notes

The banking evaluation required an enhancement of the label polarity. The custom long-horizon security predicate returns true when the trajectory is secure, so attack success is its negation. AgentDojo’s banking predicate uses the opposite convention, returning true when an attack succeeds. Thus, applying the long-horizon conversion inverted the banking labels. We modified this before generating the reported results and verified it manually. For manual verification, a refusal without a `send_money` call was marked successful, while a transfer to the attacker was marked unsuccessful.

Table 6 gives the final trace-level rates. This substantially changes the interpretation: the Anthropic models are robust in both domains, whereas the evaluated OpenAI models exhibit much higher attack success.

Scope notes. Both domains use AgentDojo’s execution and logging infrastructure, so the cross-domain result does not establish generalization to independently built agent runtimes. The evaluated

Table 5: Decoupling by model. LH uses the full graph; banking uses the conservative graph. (A dash indicates that the model was not evaluated.)

Model	LH pairs	LH rate	Bank pairs	Bank rate
GPT-4o-mini	127	91.3%	99	45.5%
GPT-4o	147	98.0%	—	—
Haiku 4.5	91	100.0%	54	18.5%
Sonnet 4.6	89	100.0%	78	29.5%

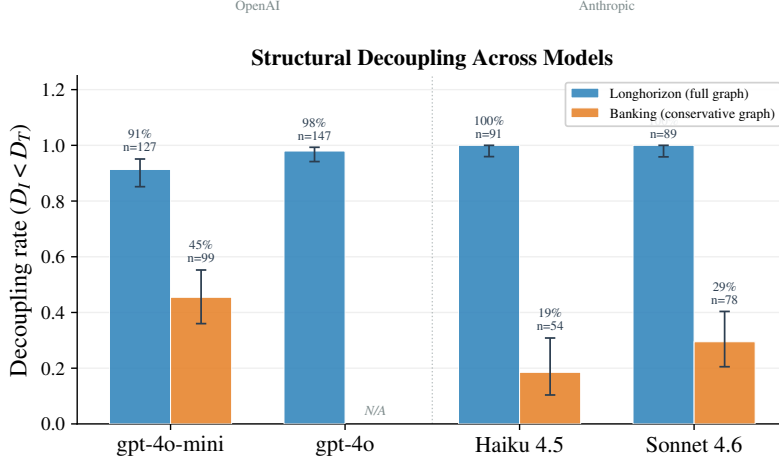


Figure 6: Structural decoupling across the four evaluated models. Error bars are Wilson 95% confidence intervals.

Table 6: Corrected attack success rates from evaluable injected traces.

Suite	Model	Successful / traces	ASR
Long-horizon	GPT-4o-mini	45/90	50.0%
Long-horizon	GPT-4o	62/90	68.9%
Long-horizon	Haiku 4.5	1/90	1.1%
Long-horizon	Sonnet 4.6	0/90	0.0%
Banking	GPT-4o-mini	40/54	74.1%
Banking	Haiku 4.5	0/141	0.0%
Banking	Sonnet 4.6	1/144	0.7%

Table 7: Additional decisions blocked relative to the sequence-only gate.

k	Full graph		Conservative	
	Attack	Benign	Attack	Benign
2	+5	+0	+5	+0
3	+5	+0	+5	+0
5	+23	+3	+7	+1
8	+48	+15	+21	+5
10	+71	+34	+36	+14

threat model is indirect prompt injection through tool outputs. Richer provenance, longer deployments, independent frameworks, and prospective threshold selection remain important future implementations.

D Full Gate Threshold Sweep

Table 7 reports all evaluated thresholds. At $k = 2$ and $k = 3$, the full and conservative graphs agree and add no benign blocks. Larger thresholds increase attack coverage but also impose a growing benign cost.

E Graph Construction Procedure

This section spells out the measurement pipeline at the level needed to reimplement it. Each trajectory is first normalized into an ordered list of events. Every event receives a stable identifier and retains its event type, tool name, arguments, returned identifiers, state key, source identifiers, and position in the trace. No free-text similarity or model judgment is used.

Table 8: Core fields in the pair-level analysis table.

Field	Meaning
run_id	Stable trajectory identifier.
suite, backend, model_id	Domain and model provenance.
task_id, attack_id	User task and injection-task identifiers.
injection_id, sink_id	Event identifiers defining the measured pair.
d_t	Shortest-path length in the sequence graph.
d_i_full, d_i_con	Influence distances under full and conservative graphs.
gap_full, gap_con	Corresponding values of $D_T - D_I$.
attack_success, task_success	Corrected security and utility outcomes.
n_events	Length of the normalized trajectory.
influence_edge_types	Edge classes appearing on the selected influence path.

For each trajectory, graph construction proceeds as follows:

1. Create one node for every normalized event and connect consecutive nodes with sequence edges.
2. Join each tool call to its recorded result. This edge class is included only in the full graph.
3. Index writes by state key and join each later read to the preceding write or writes recorded for that key.
4. Index emitted typed identifiers and join a later event when its recorded `source_ids` field contains an exact identifier match.
5. Join tool calls emitted in the same assistant batch to their shared decision event.
6. For every injection–sink pair, compute the shortest-path length in the sequence graph and in both influence-graph variants. Store D_T , D_I , and $G = D_T - D_I$ together with trace-level outcome labels.

The conservative graph contains the identical pipeline except that it does not perform step 2. All 4 kinds of graphs, having identical sequence edges, show that the sequence graph is spanned by the influence graphs, thus $D_I \leq D_T$ is a construction invariant, and useful for us to implement. Whenever $D_I > D_T$, the graph is invalid or index is wrong.

F Measurement Record Summary

Table 8 lists the fields of the pair-level analysis table, which holds one row per comparable injection–sink pair; all reported aggregates are reproducible from it.

G Complete Decoupling Statistics

Table 9 expands the abbreviated main-text result. It reports both graph variants for all long-horizon backends, as well as the conservative banking graph used for cross-domain comparison.

The model-level rows provide clarity to two different aspects. First, there is no attack success that is responsible for long-horizon decoupling; for instance, while Anthropic models have a near-zero attack success, their decoupling value is still very high. In addition, it is possible to see that the level of banking decoupling depends on infrastructure, which means that, for instance, the Haiku banking traces are the shortest ones with the lowest decoupling levels. On the contrary, GPT-4o-mini produces longer traces, offering more opportunities for provenance edges to shorten the route.

H Statistical and Reproducibility Details

Wilson intervals are used for proportions because several rates lie near zero or one. The E2 logistic regression remains restricted to the 274 OpenAI pairs: adding backends with zero or nearly zero successes causes complete or near-complete separation under ordinary maximum likelihood. We therefore avoid presenting unstable pooled coefficients as if they were comparable estimates. The

Table 9: Complete per-model decoupling summary. Medians refer to the pair-level distributions.

Suite/model	Graph	Pairs	Dec.	Rate	Med. D_T	Med. D_I	Med. G
LH GPT-4o-mini	Full	127	116	91.3%	23	8	12
LH GPT-4o-mini	Conservative	127	115	90.6%	23	9	9
LH GPT-4o	Full	147	144	98.0%	24	14	7
LH GPT-4o	Conservative	147	118	80.3%	24	17	2
LH Haiku 4.5	Full	91	91	100.0%	23	14	9
LH Haiku 4.5	Conservative	91	91	100.0%	23	18	4
LH Sonnet 4.6	Full	89	89	100.0%	23	14	8
LH Sonnet 4.6	Conservative	89	89	100.0%	23	18	2
LH all	Full	454	440	96.9%	23	12	9
LH all	Conservative	454	413	91.0%	23	16.5	3
Bank GPT-4o-mini	Conservative	99	45	45.5%	6	6	0
Bank Haiku 4.5	Conservative	54	10	18.5%	2	2	0
Bank Sonnet 4.6	Conservative	78	23	29.5%	5	5	0
Bank all	Conservative	231	78	33.8%	5	5	0

regression uses the full-graph gap and controls for sequence distance, a binary delayed-attack indicator, and backend.

The gate evaluation is a paired counterfactual replay. For a fixed threshold, the sequence and influence gates see the same recorded decision. Since $\mathcal{B}_T(k) \subseteq \mathcal{B}_I(k)$ analytically, discordance can occur only when the influence gate blocks an additional action. We report those additional actions separately for attack and benign decisions rather than claiming an unconstrained head-to-head win rate.

Code availability. The code required to reproduce the analyses and figures will be released publicly upon acceptance.