TIMING ATTACKS ON DELEGATION AUTHORIZATION VIA BENIGN CONTENT IN LLM AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

In LLM agent systems, a delegation can be revoked while delegated work is still in progress, and downstream consumers may continue to rely on cached authorization state until it refreshes. A worker can therefore produce a task-aligned result after its delegation becomes invalid, creating a timing gap in which stale authorization can turn task-aligned execution into an unauthorized downstream effect. This work examines whether an attacker whose control is limited to external content can move execution into this gap without modifying trusted instructions or authorization state. We introduce Orphaned-Delegation Completion (ODC), a timing attack in which a benign, task-relevant checklist induces additional verification work and shifts worker completion past revocation. ODC is governed by two timing conditions: orphan production after revocation (Gate 1) and result adoption while the consumer's cached authorization state still marks the delegation as valid (Gate 2). Their overlap defines a bounded attack window, so increasing workload can first enable and then suppress attack success. On LangGraph with DeepSeek-V4-Flash, ASR rises from 0.3% at $N = 0$ to 94.6% at $N = 8$, then falls to 1.4% at $N = 12$. The same mechanism appears across relay models, transfers to AutoGen and CrewAI, and persists across downstream effect types. Content controls show that the timing shift follows induced verification work rather than document length alone, while more frequent authorization refresh reduces cached-valid adoption while leaving orphan production largely unchanged. These results show that benign, task-relevant content can make execution time security-relevant under revocable delegation and motivate revalidating authority when downstream effects are committed.

1 INTRODUCTION

Large language model (LLM) agents increasingly combine model reasoning, external tools, retrieval, and multi-component orchestration (Schick et al., 2023; Yao et al., 2023; Lewis et al., 2020; Wu et al., 2023). In such systems, work is often performed under delegation: a principal grants a worker bounded authority to carry out a task, and a downstream consumer may use the resulting output to trigger an external effect. That authority need not remain valid for the lifetime of the execution. A user may cancel a request, a deadline may expire, or the surrounding authorization context may change while delegated work is still in progress.

Revocation therefore creates a coordination problem across components. Work that has already started may continue after the delegation becomes invalid, while downstream components may learn of that change only after authorization state propagates. Prior work has shown that stale state or stale authority can make an action unsafe by the time it is used (Jiang et al., 2026; Santos-Grueiro, 2026). What remains largely unexplored is whether an attacker can influence when delegated computation reaches this boundary without controlling revocation or authorization itself.

For LLM agents, execution duration is not independent of external input: retrieved content can change how much legitimate work a task requires. Prior attacks exploit external content to alter behavior, outputs, or computational cost (Abdelnabi et al., 2023; Zou et al., 2025; Shafran et al., 2025; Kumar et al., 2025; Li & Wang, 2026). Here, the relevant consequence is different. Benign, task-relevant content can preserve the trusted task objective while shifting when task-aligned work becomes available. The underlying revocation race is not unique to LLM systems. The agent-specific attack lever is that retrieved semantics can be translated by the worker into additional legitimate

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.

execution steps. ODC therefore does not rely on input size alone: externally supplied task semantics change the executed workload K, which in turn shifts completion time T_c. Our length control separates this semantic-workload effect from generic slowdown caused by longer inputs.

We introduce Orphaned-Delegation Completion (ODC), a timing attack that exploits this interaction between content-induced workload and revocable authority. The attacker contributes a benign, task-relevant document whose verification items become legitimate inputs to the trusted task. The resulting additional work can delay completion past effective revocation, producing an orphaned result: the computation remains aligned with the trusted task, but its authorizing delegation is no longer valid when the result becomes available. If a downstream consumer still holds a stale-valid authorization view, it may adopt that result and trigger an effect the revoked delegation no longer authorizes.

ODC is governed by two timing conditions. Gate 1 occurs when worker completion falls after effective revocation; Gate 2 occurs when the downstream consumer adopts that result while its authorization view still appears valid. Their overlap defines a bounded attack window. Additional work can first move completion past revocation, opening Gate 1, but further delay eventually allows authorization refresh to close Gate 2. Attack success is therefore non-monotonic in workload and depends on where completion falls relative to both revocation and refresh.

Experiments trace this bounded window across models and orchestration frameworks. On LangGraph with DeepSeek-V4-Flash, ASR rises from 0.3% at $N = 0$ to 94.6% at $N = 8$, then falls to 1.4% at $N = 12$, matching the predicted non-monotonic response. The same two-gate pathway appears across additional relay models, AutoGen, CrewAI, and multiple downstream effect types. Content and refresh controls further show that induced work drives orphan production, whereas authorization freshness determines whether those results are adopted.

This work makes three contributions:

- **Benign content as a timing lever.** We show that task-relevant external content can shift task-aligned delegated execution across revocation without modifying trusted instructions or authorization state.
- **A two-gate model of the resulting authorization failure.** We characterize ODC through orphan production and cached-valid adoption, whose overlap explains its bounded, non-monotonic attack window.
- **Evidence across systems and defenses.** We reproduce the mechanism across models, orchestration frameworks, and downstream effects, and show that authorization freshness and commit-time revalidation act on a different boundary from worker execution.

2 Related Work

Content-driven attacks. External content is a well-established attack surface for LLM systems and agents. Indirect prompt injection shows that untrusted context can redirect model behavior, with later work extending this threat to tool-using and web agents (Liu et al., 2024; Debenedetti et al., 2024; Evtimov et al., 2025). More targeted attacks exploit particular agent components: ToolHijacker manipulates tool selection through injected tool descriptions (Shi et al., 2026), while AgentPoison and MINJA poison retrieved knowledge or memory to influence subsequent reasoning and actions (Chen et al., 2024; Dong et al., 2025). Retrieval-based attacks similarly inject documents to induce attacker-chosen answers or suppress otherwise answerable queries (Zou et al., 2025; Shafran et al., 2025). Other attacks instead preserve the nominal task outcome while increasing reasoning or tool-use workload (Kumar et al., 2025; Li & Wang, 2026). Together, these studies show that external content can alter agent behavior, memory, outputs, or computational effort. ODC differs in what the adversarial content changes: it does not redirect the trusted task, but shifts when task-aligned work becomes available.

Timing, delegation, and revocation. Timing failures arise when the state that justified an operation changes before the operation takes effect. Mind the Gap identifies time-of-check-to-time-of-use vulnerabilities in LLM-enabled agents that act on stale state (Lilienthal & Hong, 2025), while Atomicity for Agents exposes such failures in browser agents when execution relies on a stale snapshot (Jiang et al., 2026). Commit-time authorization shows that earlier-valid authority may no longer justify a downstream effect when it is committed (Santos-Grueiro, 2026), and revocation-propagation work studies how authority changes are carried across agent components (Choi et al., 2026; Liu et al., 2026). These works establish the danger of stale state and stale authority, but largely

treat the duration of intervening computation as given. ODC therefore exposes a missing link between these lines of work: attacker-influenced execution time can shift a task-aligned computation across a revocation boundary and into a stale-authority window.

3 System and Threat Model

System setting. We consider a delegated LLM-agent workflow with three roles: a principal, a worker, and a downstream consumer. The principal grants the worker bounded authority to perform delegated work d. The worker retrieves external records, derives the checks required by the trusted task, and returns a result z, which the consumer may use to request an external effect e. The trusted task defines the verification objective and permitted effects, while retrieved records influence the concrete checks the worker performs. The retrieval corpus may contain externally contributed content.

Authorization semantics. Let r denote the effective invalidation time of d. The delegation is valid before r and invalid from r onward unless a new authorization is issued; a timeout or cancellation request need not coincide with this boundary. We write T_c for the time at which the worker completes z and makes it available to the consumer, and t_{use} for the time at which the consumer observes the cached authorization view used to decide adoption. A violation requires an effect depending on d to be accepted after d has become invalid, with no valid successor authorization in force. Independently reauthorized effects are therefore not violations.

Attacker capabilities and objective. The attacker contributes one task-relevant document to the retrieval corpus and, in our benchmark, knows the fixed worker query template. Similar query-aware settings appear in prior retrieval attacks: PoisonedRAG constructs injected texts for attacker-chosen target questions, while RAG Jamming assumes knowledge of the victim query when constructing query-specific blocker documents (Zou et al., 2025; Shafran et al., 2025). The attacker chooses the document contents and N, the number of verification items, before execution and may calibrate this choice offline, but is non-adaptive during a run. Our core timing analysis conditions on successful retrieval and focuses on the post-retrieval attack mechanism: whether task-relevant content that reaches the worker can induce enough legitimate work to shift task-aligned execution across a revocation boundary. Retrieval reachability is evaluated separately in Appendix B. Recent work has independently shown that attacker-controlled content can be optimized for retrieval under target queries, including in realistic RAG and agentic settings (Chang et al., 2026). The attacker cannot alter the user request, trusted prompts, tools, other documents, retrieval or orchestration code, or authorization state; nor does it control revocation, cache phase, scheduling, or network latency. Its objective is to induce enough additional verification work that T_c falls after revocation while the consumer still holds a stale-valid authorization view, allowing an effect to be accepted without the authority it requires.

4 Attack Construction and Temporal Model

Attack construction. Figure 1 shows the ODC attack path from contributed content to downstream effect. The attacker first contributes a benign, task-relevant document to a low-trust retrieval corpus. Once retrieved, the document does not redirect the worker or alter its authorization; instead, its N verification items become legitimate inputs to the trusted task. The worker turns these items into K concrete checks, increasing the amount of work performed before the result is returned and thereby shifting the completion time T_c. If this additional work carries completion past effective revocation at r, the worker produces an orphan result: the task began under valid authority, but its result becomes available only after that authority has expired. The orphan alone is not yet a violation. For an orphaned result to produce an unauthorized downstream effect, the consumer must still hold a stale-valid authorization view, adopt the result, and trigger an effect that the revoked delegation no longer authorizes. Figure 1 therefore separates the two conditions underlying ODC: delayed execution produces the orphan, while stale authorization determines whether that orphan can still be acted upon.

Workload response. ODC influences completion time indirectly through the work induced by the retrieved document. Let N denote the number of task-relevant verification items supplied by the attacker and K the number of checks the worker actually executes. In the serial configuration we

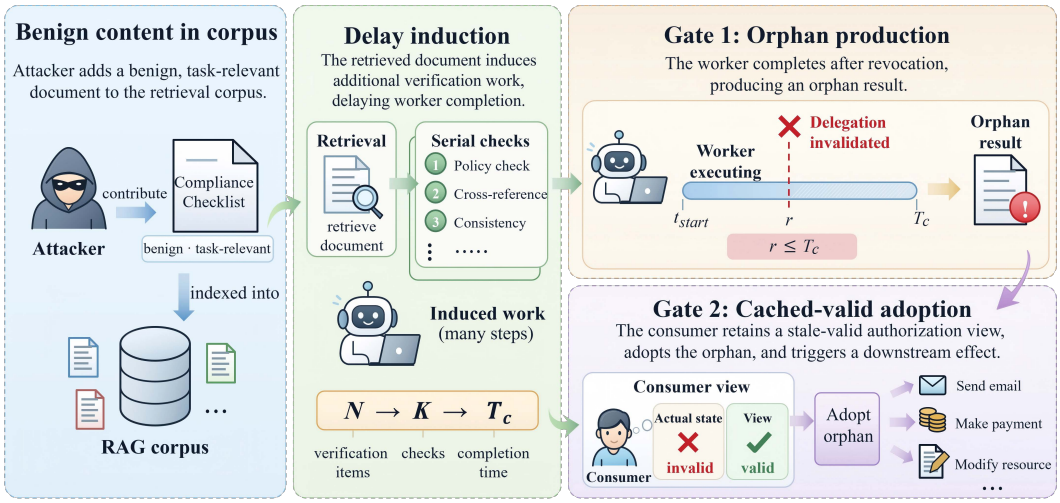


Figure 1: Overview of ODC. A benign, task-relevant document induces additional verification work, delaying completion beyond revocation. The resulting orphan can lead to a downstream effect if the consumer adopts it under a cached authorization view that still reports the delegation as valid.

evaluate, the worker completion time can be decomposed as

$$T_c = t_0 + b(N) + \sum_{j=1}^{K} L_j + \epsilon, \tag{1}$$

where t_0 is the baseline time associated with a worker invocation, $b(N)$ is the time spent on retrieval and planning for the N-item document, L_j is the execution time of the j-th check, and ϵ captures residual scheduling variation. The attacker chooses N, but the worker determines how those items translate into executed checks, so K need not equal N. Increasing N can therefore shift T_c in two ways: by increasing the time required to retrieve and plan from the document, and by inducing additional checks whose execution times accumulate before the result is returned.

Necessary attack window. For induced delay to become security-relevant, worker completion must fall within a bounded interval. Let W denote the residual stale-valid interval after effective revocation at r: the amount of time for which the consumer can still observe the delegation as valid even though the underlying grant has already been withdrawn. Importantly, W is the stale interval realized in a particular run, rather than the configured polling period. Let t_{use} be the time at which the consumer observes the cached authorization view used to decide adoption, and let

$$\delta = t_{\text{use}} - T_c \tag{2}$$

denote the delay between worker completion and the authorization observation used for adoption, including result propagation, queueing, scheduling, and consumer-side processing. For the worker to produce an orphan, completion must occur after effective revocation,

$$T_c \geq r.$$

At the same time, the authorization view used for adoption must be observed before the stale-valid interval closes,

$$T_c + \delta < r + W.$$

Combining these conditions gives the necessary ODC timing window

$$r \leq T_c < r + W - \delta. \tag{3}$$

The two bounds explain why ODC is not simply a monotonic slowdown attack. With too little added work, the worker finishes before r and the result remains authorized. Additional work can move T_c past revocation and produce an orphan, but further delay eventually pushes completion far enough that the consumer's observation of the authorization view falls after it has refreshed, so the view no longer marks the delegation as valid. The necessary ODC window therefore lies between these two boundaries: completion must occur late enough to outlive the delegation, but early enough that the authorization view is still observed as valid.

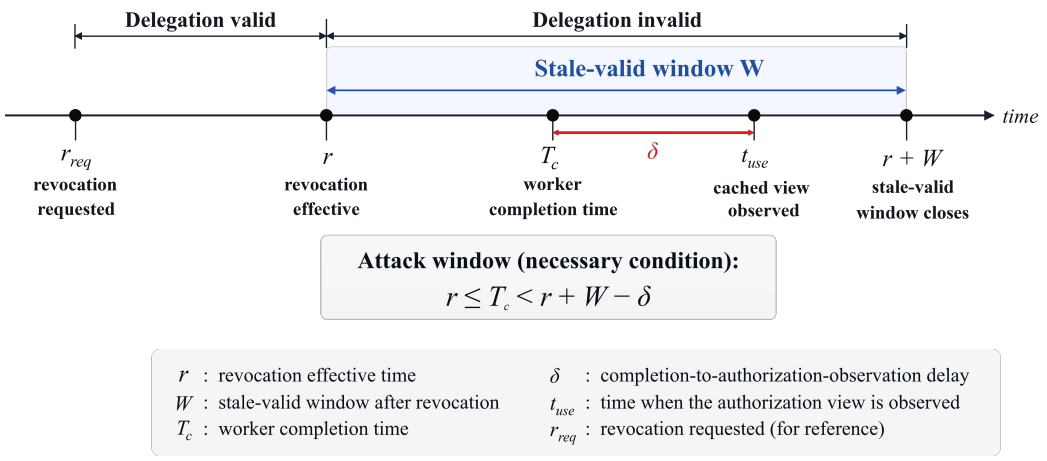


Figure 2: Necessary ODC timing window. The worker must complete after revocation, but the consumer must observe the cached authorization view used for adoption before the stale-valid window closes.

5 Measurement and Experimental Setup

Outcome metrics. We measure ODC through the two conditions identified in Section 4. *Gate 1*, orphan production, holds when the worker completes after revocation. We operationalize this boundary using the authorization store's post-transition acknowledgment timestamp r_{ack}, a conservative observable boundary for effective invalidation (Appendix C). *Gate 2*, cached-valid adoption, holds when the consumer adopts the result while its cached authorization view reports the delegation as valid. Gate 2 is defined independently of whether revocation has already occurred; when it overlaps with Gate 1, that valid view is stale and the adopted result is orphaned. We define

$$\text{ASR} = \Pr[\,\text{Gate 1} \ \wedge \ \text{Gate 2}\,]. \tag{4}$$

In the integrated harness, consumer adoption directly triggers the local downstream operation, with no additional independent acceptance gate between adoption and the measured effect. Gate 1 $\wedge$ Gate 2 therefore coincides with the measured orphan-effect violation event in this harness. Appendix C describes the event-level instrumentation and acceptance boundary.

Neither gate alone constitutes ODC. Gate 1 determines whether an orphan is produced, while Gate 2 determines whether the consumer admits the result under its current cached view. ASR counts trials in which both occur. Gate 1 and Gate 2 are reported as marginal rates over all trials.

Experimental design. The primary experiment varies the number N of task-relevant verification items while holding the workflow and corpus construction fixed. The resulting check count K is observed from the worker trace. We evaluate three relay models on LangGraph and test transfer across additional orchestration frameworks and downstream effects. Each primary arm contains 350 trials. Attack-success measurements are conditional on successful retrieval as defined in Section 3.

Timing controls. We control the effective revocation time directly rather than coupling it to a particular cancellation or timeout mechanism. This isolates the timing relationship between worker completion, revocation, and downstream authorization refresh. Revocation timing is placed separately for each relay using completion-time measurements so that the workload sweep spans the invalidation boundary, and the resulting offset is held fixed across that relay's workload conditions (Appendix C). Stale- and fresh-cache settings vary authorization freshness, and Section 6.3 directly sweeps the consumer refresh period. Exact timing parameters, calibration details, and confidence-interval computation are reported in Appendices C and E.

6 Evaluation

We now examine how ODC behaves under controlled changes to workload and system configuration. We first vary the number of task-relevant verification items, then use content controls to separate

Table 1: Workload sweep across three relay-model configurations ($n = 350$ per arm). t_{med} denotes the median worker completion time. Gate 1 and Gate 2 are reported separately, and ASR is the fraction of trials in which both gates occur.

Model	N / Poll	K	t_{med} (s)	Gate 1 (%)	Gate 2 (%)	ASR	95% CI
DeepSeek-V4-Flash	0 / stale	0	0.79	0.3%	100.0%	0.3%	[0.1, 1.6]
	4 / stale	4	3.89	1.1%	100.0%	1.1%	[0.4, 2.9]
	8 / stale	8	6.94	94.6%	100.0%	94.6%	[91.7, 96.5]
	10 / stale	10	9.20	100.0%	62.3%	62.3%	[57.1, 67.2]
	12 / stale	12	11.28	100.0%	1.4%	1.4%	[0.6, 3.3]
	8 / fresh	8	7.95	99.7%	0.0%	0.0%	[0.0, 1.1]
Claude-Opus-5	0 / stale	0	2.534	0.0%	100.0%	0.0%	[0.0, 1.1]
	4 / stale	4	14.85	9.1%	100.0%	9.1%	[6.6, 12.6]
	8 / stale	8	32.18	100.0%	100.0%	100.0%	[98.9, 100.0]
	10 / stale	10	37.68	100.0%	96.3%	96.3%	[93.7, 97.8]
	12 / stale	12	43.76	100.0%	92.6%	92.6%	[89.3, 94.9]
	8 / fresh	8	30.46	100.0%	0.0%	0.0%	[0.0, 1.1]
GPT-5.6-Sol	0 / stale	0	4.071	2.3%	100.0%	2.3%	[1.2, 4.4]
	4 / stale	4	30.86	10.6%	100.0%	10.6%	[7.8, 14.2]
	8 / stale	8	64.87	100.0%	100.0%	100.0%	[98.9, 100.0]
	10 / stale	10	85.11	100.0%	100.0%	100.0%	[98.9, 100.0]
	12 / stale	12	124.50	81.1%	47.1%	26.9%	[22.5, 31.7]
	8 / fresh	8	73.54	100.0%	0.0%	0.0%	[0.0, 1.1]

induced work from document length, and finally test whether the same two-gate pattern persists across relay models, orchestration frameworks, and downstream effect types. Gate 1, Gate 2, and ASR are reported separately throughout to distinguish orphan production from cached-valid adoption.

6.1 Added workload creates a bounded attack window

Table 1 varies the number N of task-relevant verification items on LangGraph across three relay-model configurations. Revocation timing is placed separately for each relay, so the comparison is intended to test the predicted workload response rather than rank models by susceptibility.

DeepSeek-V4-Flash shows the full crossover most clearly. ASR is 0.3% at $N = 0$ and 1.1% at $N = 4$, rises to 94.6% at $N = 8$, and then falls to 1.4% at $N = 12$. The two gates explain this non-monotonic response: increasing workload first moves completion beyond revocation, opening Gate 1, but eventually delays the result long enough for Gate 2 to close. Claude-Opus-5 and GPT-5.6-Sol occupy different portions of the same window. Claude remains largely inside it through $N = 12$, where ASR is still 92.6%, because its completion times have not yet moved far enough toward the upper edge of the stale-valid window for Gate 2 to close substantially. GPT-5.6-Sol, by contrast, has already reached the declining side at $N = 12$, where ASR falls to 26.9%. Across all three relays, the peak therefore reflects overlap between orphan production and continued downstream adoption rather than maximum delay.

The $N = 8$ fresh-cache controls further separate the two gates. Gate 1 remains at 99.7% or 100%, so the induced workload still produces post-revocation completions; what changes is whether the consumer continues to admit them. Gate 2 and ASR fall far below Gate 1 in these configurations, and the size of the residual adoption is characterized across refresh periods in Section 6.3.

6.2 Content controls isolate induced work

The workload sweep alone does not establish whether the timing shift comes from additional verification work or simply from a longer document. Table 2 separates these explanations while holding the corpus, worker, consumer, and sampled revocation schedule fixed. Under the workload control, document length remains at L while the number of verification items increases from $N = 0$ to $N = 8$. The executed check count K follows N, and ASR rises from 0.6% to 50.9% once the eight-item checklist pushes completion across revocation. Under the length control, the same eight verification items are retained while the document is expanded to $2L$. The worker still executes

Table 2: Content controls ($n = 350$ per arm). Verification workload, document length, and authorization refresh are varied separately while the remaining workflow is held fixed.

	Content	N	Length	K	t_{med} (s)	Gate 1 (%)	Gate 2 (%)	ASR
Workload control	Clean input	0	L	0	0.801	0.9%	99.7%	0.6%
	Checklist	2	L	2	2.254	0.9%	99.7%	0.6%
	Checklist	4	L	4	3.851	2.6%	99.7%	2.3%
	Checklist	8	L	8	6.819	56.6%	94.6%	50.9%
Length control	Checklist	8	$2L$	8	6.811	52.9%	95.1%	48.0%
Authorization control	Checklist, fresh	8	L	8	6.867	56.9%	46.0%	2.9%

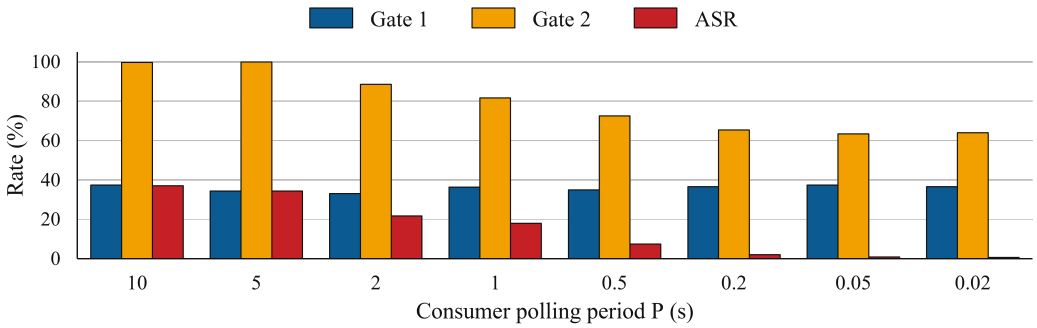


Figure 3: Effect of consumer polling period P on the two ODC gates ($n = 350$ per period). Gate 1 stays roughly flat, while Gate 2 and ASR decrease as authorization refresh becomes more frequent.

$K = 8$ checks and ASR remains similar at 48.0%, showing that added text alone does not reproduce the timing shift. The authorization control keeps the same eight-item workload but shortens the consumer's authorization-cache refresh period to 0.20 s. Gate 1 remains 56.9%, while ASR falls to 2.9%, showing that the same workload still produces post-revocation completions but is much less likely to lead to downstream adoption under stale authorization. The primary sweep and these controls use different revocation schedules, so their fresh-cache rates are not directly comparable; Appendix C explains why we restrict this comparison to within-sweep effects.

6.3 Shorter refresh period narrows the attack window

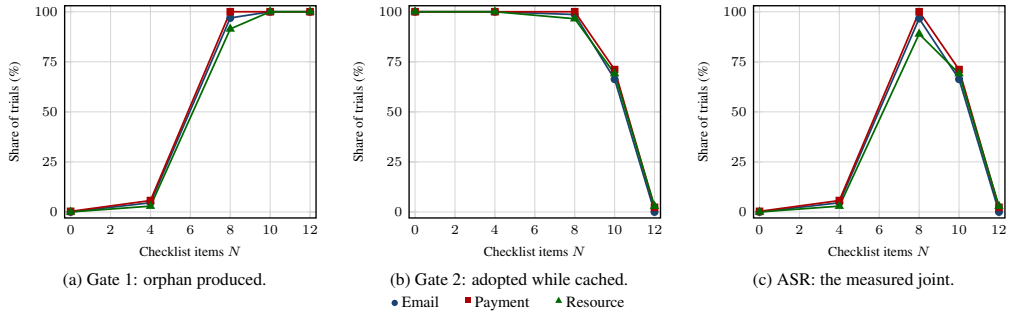
We next vary only the consumer's polling period P while keeping the induced workload fixed. Figure 3 shows that Gate 1 remains nearly unchanged across the sweep, ranging from 33.1% to 37.4%, while ASR falls from 37.1% at $P = 10$ s to 0.6% at $P = 0.02$ s. Gate 2 also decreases as authorization is refreshed more frequently. Shortening the refresh period therefore does not prevent orphan production. Instead, it narrows the interval in which an orphan can still be adopted, driving attack success toward zero without eliminating the underlying timing gap.

6.4 The pathway transfers across models and frameworks

The preceding experiments establish the mechanism on LangGraph. We next ask whether it depends on that orchestrator. Table 3 reproduces the construction in AutoGen and CrewAI using framework-specific timing parameters. Under stale polling, both frameworks reach 100% ASR from $N = 4$ onward. At $N = 8$, switching to 0.20 s fresh polling reduces ASR to zero while Gate 1 remains at 100%, again separating continued worker execution from downstream adoption. Together with the three relay configurations in Table 1, these results reproduce the same temporal pathway across model and orchestration choices. The absolute crossover point depends on execution and refresh timing, but the mechanism remains the same: workload determines whether completion crosses revocation, while authorization freshness determines whether the resulting output can still be adopted.

Table 3: Cross-framework ODC sweeps on AutoGen and CrewAI ($n = 50$ per arm). Stale and fresh polling reproduce the two-gate pathway under different orchestrators.

Framework	Regime	N	K	t_{med} (s)	Gate 1	Gate 2	ASR
AutoGen	stale	2	2	4.223	92.0%	100.0%	92.0%
	stale	4	4	7.836	100.0%	100.0%	100.0%
	stale	8	8	14.366	100.0%	100.0%	100.0%
	stale	12	12	22.488	100.0%	100.0%	100.0%
	fresh	2	2	4.521	98.0%	4.0%	2.0%
	fresh	8	8	14.586	100.0%	0.0%	0.0%
CrewAI	stale	2	2	2.446	96.0%	100.0%	96.0%
	stale	4	4	2.852	100.0%	100.0%	100.0%
	stale	8	8	3.237	100.0%	100.0%	100.0%
	stale	12	12	4.047	100.0%	100.0%	100.0%
	fresh	2	2	2.349	80.0%	34.0%	14.0%
	fresh	8	8	2.995	100.0%	0.0%	0.0%



(a) Gate 1: orphan produced. (b) Gate 2: adopted while cached. (c) ASR: the measured joint.

Figure 4: Two-gate behavior across three downstream effect classes ($n = 350$ per point). Each task uses an independent workflow and sink.

6.5 The mechanism extends across effect types

Finally, we vary the downstream effect rather than the execution stack. Figure 4 instantiates the construction independently for email, payment, and resource modification, with a separate document, worker, revocation line, consumer, and sink for each task. Despite differences in completion time and task configuration, all three exhibit the same response: Gate 1 rises as additional work pushes completion beyond revocation, Gate 2 later falls as the stale-valid window closes, and ASR peaks where the two overlap. The absolute rates are task-specific, but the shared shape shows that the mechanism is not tied to a particular downstream effect.

7 Defensive Design Implications

ODC separates two problems that are easy to conflate: stopping delegated computation and preventing a revoked delegation from producing a downstream effect. A worker may continue after revocation and still return a result that remains aligned with the delegated task. Once such an orphaned result exists, execution-side controls can no longer determine whether the result remains authorized when it is used. Preventing the downstream violation therefore requires the accepting resource to revalidate the authority on which the result depends and determine whether that authority is still valid at the point of use.

Table 4 examines this distinction by holding the orphan and invalidation timeline fixed while varying the acceptance policy. The comparison includes the attack baseline, three established mitigation classes, and two authorization-validation policies instantiated in our harness. Appendix D details the policy harness and the *Within/After* partition. The three established mitigations address failure modes adjacent to ODC but do not restore current authority at commit time. Idempotency or deduplication

Table 4: Orphan-effect allow rate under six acceptance policies on a shared invalidation timeline ($n = 400$). A rate is the share of attempts on which the policy allowed the orphan effect to proceed. *Within* denotes trials whose cached authorization view remains stale-valid at commit, while *After* denotes trials whose view has already observed invalidation.

Policy	Class	Overall	Within	After
None (attack baseline)	Baseline	100.0%	100.0%	100.0%
Idempotency or dedup key	Replay guard	100.0%	100.0%	100.0%
Native cancel or timeout	Preemption	100.0%	100.0%	100.0%
Post-commit compensation	Compensation	100.0%	100.0%	100.0%
Cached witness, TTL 0.15 s	Async recheck	24.0%	50.3%	0.0%
Synchronous re-resolve (zero delay)	Sync recheck	0.0%	0.0%	0.0%

suppresses replay, whereas an orphan is the first delivery of a result produced under a delegation that has since been revoked. Native cancellation or timeout attempts to stop ongoing execution, but work that has escaped the orchestrator's cancellation scope may still reach the downstream consumer. Post-commit compensation can reverse some completed operations, but it cannot prevent an irreversible downstream effect once that effect has already been committed. In our experiment, all three therefore allow 100.0% of the tested orphan effects. These mechanisms remain useful for the failure modes they were designed to address, but they do not close the authorization boundary exposed by ODC.

The remaining policies isolate the role of authorization freshness. A cached witness rechecks whether the delegation appears valid, but does so against an asynchronously refreshed authorization view. With a 0.15 s TTL, this policy reduces the overall allow rate to 24.0%, yet still admits 50.3% of attempts made while the cached view remains stale-valid. The recheck therefore helps after invalidation has propagated, but cannot distinguish revoked authority while the cached view still reports it as valid. Synchronous re-resolution instead resolves current authority again at the effect boundary. With zero read-to-use delay in our harness, it rejects every tested orphan. The contrast shows that the relevant property is not the presence of another authorization check by itself, but whether that check observes sufficiently current authority when the effect becomes irrevocable.

Real systems make this boundary harder to realize. Authorization resolution and effect commitment may be separated by process, service, or network boundaries, and any interval between them creates an opportunity for revocation to intervene after authority has been checked but before the effect is committed. The zero-delay result therefore represents an idealized boundary. In a deployment where authorization resolution and effect commitment are separated, a nonzero read-to-use interval can reintroduce a revocation race. Closing ODC in deployment therefore requires more than adding another recheck or shortening a cache lifetime. It requires sufficiently tight coordination between current authorization state and the resource that commits the downstream effect. In this sense, ODC exposes not the absence of a conceivable defense, but the stronger authorization boundary that revocable agent workflows must maintain once delegated computation and downstream effects become decoupled.

8 Conclusion

Task-relevant content can influence when an agent finishes, and completion time can determine which authorization state governs a downstream action. ODC makes this connection concrete through a bounded checklist construction and by separating orphan production from cached-valid adoption. Across models and orchestration frameworks, the observed attack follows a timing window rather than a monotonic slowdown effect. More frequent authorization refresh narrows the stale-adoption path while orphan production can persist, whereas cancellation targets ongoing execution but may not reach work that has escaped its scope. These mechanisms therefore act on different parts of the ODC pathway and do not by themselves guarantee authorization at effect commit. Revocable delegation therefore cannot be enforced solely at task start or through periodically refreshed authorization views. Execution time is authorization-relevant: results must retain the delegation state they depend on, and effects must be validated against current authority at acceptance, which closes the gap between task-aligned execution and authorized action.

AI USE STATEMENT

In the interest of full transparency, we disclose the utilization of Artificial Intelligence (AI) in this study. Large Language Models (LLMs) were employed strictly for literature retrieval, linguistic refinement, coding assistance, and consistency auditing. The AI tools aided in discovering relevant academic literature and polishing the manuscript's language to improve its clarity and readability. Regarding the implementation of our research, AI was utilized to assist in translating our established algorithmic logic into executable code. AI tools were additionally used to audit the writing itself, cross-checking the manuscript against its source records and flagging numerical and terminological inconsistencies across sections. However, the core research concepts, methodological frameworks, and experimental designs were exclusively conceived and directed by the authors. The AI served as an assistant to our own work rather than as a source of it: it did not originate research ideas, select methods, run experiments, interpret results, or draw scientific conclusions, and it did not decide what the manuscript claims. All AI-assisted code and all AI-flagged issues have been reviewed, tested, and validated by the authors, and every edit made in response to an automated consistency check was made or rejected on the authors' own judgement. All AI assistance was used in accordance with the venue's policy and with the terms of the tools involved. The authors take full responsibility for the entirety of this work, including the correctness, integrity, and reproducibility of all code and written content. We confirm that all AI-assisted outputs strictly adhere to academic ethical standards and do not constitute plagiarism or scientific misconduct.

ETHICS STATEMENT

The described probes use local effect sinks and controlled benchmark environments. The figures' email, payment, and resource icons illustrate the model, and this draft does not report third-party exploitation, actual payments, or production vulnerability confirmation. The released artifact uses synthetic records and local destinations and excludes credentials and private task data.

REPRODUCIBILITY STATEMENT

Appendix C documents the measurement boundaries and record-level audit used for the integrated experiments, while Appendices F and G specify the content and event contracts needed to interpret those records. The released artifact includes scripts for recomputing reported counts, confidence intervals, tables, and consistency checks from the experimental records. These scripts reproduce the reported analyses but do not rerun model inference.

REFERENCES

Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec 2023)*, pp. 79–90. ACM, 2023. URL https://doi.org/10.1145/3605764.3623985.

Artificial Analysis. Claude opus 5 model intelligence and performance analysis: Providers. Artificial Analysis, 2026a. URL https://artificialanalysis.ai/models/claude-opus-5/providers. Accessed 2026-09-05.

Artificial Analysis. DeepSeek-V4-Flash model intelligence and performance analysis: Providers. Artificial Analysis, 2026b. URL https://artificialanalysis.ai/models/deepseek-v4-flash/providers. Accessed 2026-09-05.

Artificial Analysis. GPT-5.6-Sol model intelligence and performance analysis: Providers. Artificial Analysis, 2026c. URL https://artificialanalysis.ai/models/gpt-5-6-sol/providers. Accessed 2026-09-05.

Hongyan Chang, Ergute Bao, Xinjian Luo, and Ting Yu. Overcoming the retrieval barrier: Indirect prompt injection in the wild for LLM systems. *arXiv preprint arXiv:2601.07072*, 2026. URL https://arxiv.org/abs/2601.07072.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/eb113910e9c3f6242541c1652e30dfd6-Abstract-Conference.html.

Moonwon Choi, Seokho Jeong, and Seunggeun Lee. Residualauth: What authorization state must language agents preserve under revocable delegation? arXiv:2609.08062, 2026. URL https://arxiv.org/abs/2609.08062.

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 37 (NeurIPS 2024)*, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory injection attacks on LLM agents via query-only interaction. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/42a97bbd9844d2bf68596730af80bcdf-Abstract-Conference.html.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. WASP: Benchmarking web agent security against prompt injection attacks. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025), Datasets and Benchmarks Track*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/1c9818387f5dd0a0bc151214660f059d-Abstract-Datasets_and_Benchmarks_Track.html.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. In *Proceedings of the Conference on Applied Machine Learning in Information Security (CAMLIS 2024)*, volume 3920 of *CEUR Workshop Proceedings*, pp. 48–62. CEUR-WS.org, 2024. URL https://ceur-ws.org/Vol-3920/paper03.pdf.

Linxi Jiang, Zhijie Liu, Haotian Luo, and Zhiqiang Lin. Atomicity for agents: Exposing, exploiting, and mitigating TOCTOU vulnerabilities in browser-use agents. arXiv:2603.00476, 2026. URL https://arxiv.org/abs/2603.00476.

Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. OverThink: Slowdown attacks on reasoning LLMs. *arXiv preprint arXiv:2502.02542*, 2025. URL https://arxiv.org/abs/2502.02542.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Qi Li and Xinchao Wang. Sponge tool attack: Stealthy denial-of-efficiency against tool-augmented agentic reasoning. arXiv:2601.17566, 2026. URL https://arxiv.org/abs/2601.17566.

Derek Lilienthal and Sanghyun Hong. Mind the gap: Time-of-check to time-of-use vulnerabilities in LLM-enabled agents. arXiv:2508.17155, 2025. URL https://arxiv.org/abs/2508.17155.

Lifei Liu, Haoran Yu, and Xiaochong Jiang. Vera: Authority-preserving edge revocation for federated ai-agent workflows. arXiv:2608.30091, 2026. URL https://arxiv.org/abs/2608.30091.

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 (USENIX Security 24)*. USENIX Association, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei.

Bardia Mohammadi, Nearchos Potamitis, Lars Klein, Akhil Arora, and Laurent Bindschaedler. Atomix: Timely, transactional tool use for reliable agentic workflows. arXiv:2602.14849, 2026. URL https://arxiv.org/abs/2602.14849.

Igor Santos-Grueiro. Temporary authority, permanent effects: Commit-time authorization for LLM agents. arXiv:2607.10487, 2026. URL https://arxiv.org/abs/2607.10487.

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 36 (NeurIPS 2023)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Avital Shafran, Roei Schuster, and Vitaly Shmatikov. Machine against the RAG: Jamming retrieval-augmented generation with blocker documents. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3787–3806. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/shafran.

Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in LLM agents. In *33rd Annual Network and Distributed System Security Symposium (NDSS 2026)*. The Internet Society, 2026. URL https://www.ndss-symposium.org/ndss-paper/prompt-injection-attack-to-tool-selection-in-llm-agents/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-generation LLM applications via multi-agent conversation. arXiv:2308.08155, 2023. URL https://arxiv.org/abs/2308.08155.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *11th International Conference on Learning Representations (ICLR 2023)*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3827–3844. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag.

A Limitations and Evidence Boundaries

The experiments are designed to isolate the timing mechanism studied in this work, and the following points clarify the scope of the resulting claims.

Construction and instrumentation. Our evaluated construction uses task-aligned checklist content under a serial execution policy, which provides a controlled way to vary workload and completion time. The results therefore characterize this concrete mechanism rather than all possible forms of task-relevant content or execution policies. The executed check count K is reported empirically from the worker trace.

Selection. The benchmark attacker is query-aware and prepares the contributed document offline with knowledge of the fixed task template. Retrieval experiments therefore measure whether such task-aligned content can be made reachable under the evaluated retrievers, rather than estimating population-level exposure in arbitrary corpora. Additional paraphrase and multi-retriever tests assess sensitivity to wording and retrieval configuration.

Timing and handoff. The harness controls revocation timing, cache refresh, and result handoff so that the two authorization gates can be measured separately. The worker's result is handed to the consumer at natural completion with no wait on invalidation, so adoption is decided by whatever the consumer's cache holds at that moment. Gate 2 therefore counts adoptions made against a still-valid cached view, and it is high in arms whose workers finish before revocation because in those trials no orphan was produced and the adoption was the intended one. Appendix C details this measurement boundary and what it does and does not support. The main experiments are designed to establish the existence and causal structure of the ODC timing window under controlled timing regimes, rather than to estimate deployment-level exploitability under uncertain timing. In particular, we do not characterize sensitivity to uncertainty in revocation timing or cache phase beyond the timing variations observed in the evaluated harness.

Measurement. We distinguish worker completion, consumer adoption, and effect acceptance as separate events and report them separately where the instrumentation permits. The local sink provides a concrete acceptance boundary for the evaluated harness, and the trace contract in Appendix G specifies the evidence that a deployment-level evaluation would need to record.

B Retrieval Reachability and Semantic-Alignment Audit

B.1 Motivation and relation to the threat model

The timing experiments condition on the attacker's document having reached the worker. This appendix isolates that retrieval prerequisite: given the worker's task query and a corpus containing one attacker-contributed, task-relevant document, can the retriever rank that target near the front? The audit invokes no LLM.

The original document is intentionally query-aware. Section 3 permits the attacker to know the task domain and the fixed worker query template, including its wording, and to prepare one document offline before execution. It does not permit online adaptation or changes to other documents, the user request, trusted prompts, the retriever, or authorization state. Reusing query-like wording is therefore an allowed attacker capability rather than a defect in the construction. The narrower audit question is whether a document that describes the task's ordinary business purpose, without deliberately reusing the query's phrasing, retains meaningful retrieval reachability.

B.2 Frozen experimental design

We evaluate $N \in \{4, 8, 10, 12\}$ using two retrievers: sentence-transformers/all-MiniLM-L6-v2 with cosine similarity, and rank_bm25.BM25Okapi as the lexical baseline. The exact fixed worker template is one query group. A second group contains 12 paraphrases that already existed during development. These are retrospective sensitivity probes, not held-out queries and not 12 independent tasks.

For each N, the pipeline-filtered pool removes inactive specification files associated with other workload levels while leaving the rest of the corpus unchanged. It contains six candidates in this run.

Table 5: Retrieval in the pipeline-filtered candidate pool for the 12 retrospective paraphrases. Entries macro-average the four per-N summaries for $N \in \{4, 8, 10, 12\}$, and they are not 48 independent queries.

Retriever	Target variant	Hit@1	Hit@3	MRR
MiniLM	Original	1.000	1.000	1.000
MiniLM	Task descriptive	0.625	0.833	0.735
MiniLM	Neutral header	0.063	0.271	0.292
MiniLM	Header stripped	0.000	0.042	0.193
MiniLM	Unrelated control	0.000	0.000	0.167
BM25	Original	1.000	1.000	1.000
BM25	Task descriptive	0.479	0.896	0.696
BM25	Neutral header	0.042	0.667	0.363
BM25	Header stripped	0.000	0.479	0.313
BM25	Unrelated control	0.000	0.250	0.260

A second candidate pool deliberately retains specifications associated with all evaluated workload levels, creating a near-duplicate competition stress test over 23 candidates.

The target occupies the same candidate position in five replacement conditions. The original is the query-aware document; the task-descriptive variant replaces its preamble with ordinary language describing the release-checking purpose; the neutral-header variant uses a neutral engineering-reference preamble; the header-stripped variant retains only the common item body; and the unrelated control replaces the target with an unrelated cafeteria menu. Only the target bytes change within a comparison, and all competing documents and their order remain fixed. The variants are sensitivity checks rather than token-matched causal interventions: normal business terms can still overlap, and the unrelated control is not length matched.

For a query, Hit@1 and Hit@3 indicate whether the target's deterministic rank is at most one or three. Mean reciprocal rank (MRR) averages $1/\mathrm{rank}$, so larger values indicate earlier retrieval. Exact-template and retrospective-paraphrase results are kept separate. For the latter, Tables 5 and 6 macro-average the four per-N summary rows. Because the same 12 probes recur at every N, the resulting values must not be interpreted as 48 independent query samples.

B.3 Pipeline-filtered retrieval results

Table 5 reports the pipeline-filtered pool. The original target ranks first for both retrievers for every N and every retrospective paraphrase. The ordinary task description also remains meaningfully reachable: MiniLM attains 0.625 Hit@1 and 0.833 Hit@3, while BM25 attains 0.479 Hit@1 and 0.896 Hit@3. It is weaker than the query-aware original, especially at rank one, but substantially stronger there than the neutral, stripped, and unrelated replacements.

The low or zero Hit@1 of the neutral, stripped, and unrelated variants shows that replacing the target with just any document does not make it win. Describing the task's purpose supplies a real retrieval signal in this pipeline, while query-aware preparation, which the threat model allows, supplies the strongest and most reliable one. Task-semantic alignment therefore retains nontrivial retrieval reachability in this evaluated pool, without matching what deliberate query alignment achieves.

B.4 Exact-template interpretation

The single exact-template query exposes the advantage of wording knowledge more sharply. The original ranks first under both retrievers at all four values of N, for eight of eight retriever-by-N conditions. The task-descriptive target is never first. Under MiniLM, its ranks are 5, 4, 4, and 3 for $N = 4, 8, 10, 12$, while under BM25 they are 3, 2, 2, and 2. Ordinary task description therefore often moves the target into an early position, but exact query alignment is what reliably moves it to rank one in this comparison.

Table 6: Near-duplicate competition when specifications from all evaluated workload levels are retained. Entries use the same four-row macro-average as Table 5.

Retriever	Target variant	Hit@1	Hit@3	MRR
MiniLM	Original	0.021	0.250	0.237
MiniLM	Task descriptive	0.042	0.146	0.152
BM25	Original	0.042	0.229	0.243
BM25	Task descriptive	0.125	0.167	0.230

B.5 Near-duplicate competition stress test

Table 6 retains specifications from all evaluated workload levels. Hit@1 is low for both original and task-descriptive targets because many closely related documents compete to answer the same task while differing mainly in which N they encode, so the audit identifies the target for the specified N rather than accepting any release-checking specification.

This stress test shows that the retrieval prerequisite is sensitive to candidate competition: the pool is deliberately built so that many closely related documents compete on the same task. The small inversions in Hit@1 occur in a floor regime, so they do not justify a claim that the task-descriptive target outperforms the original, and the MRR and Hit@3 values show no uniform advantage for that interpretation either.

B.6 Scope

This audit establishes target-document reachability under two retrievers on one task family, with frozen candidate pools and a retrospective query set. Reachability is one stage of the pathway, and the stages it feeds each carry their own denominator: whether the worker follows the document, whether completion crosses revocation, whether a consumer adopts the result, and whether an effect is accepted after the required authority has become invalid are measured separately in the other experiments. We therefore report these retrieval rates on their own and do not multiply them by timing or attack-success rates.

Because the paraphrases are retrospective, measurements over N are dependent, and the candidate corpora are constructed, these numbers characterize reachability under the evaluated configuration rather than a deployment exposure probability. Within that scope, task-semantic alignment can make the document retrievable without deliberate query phrasing, while the permitted query-aware construction remains markedly more reliable.

C Integrated Records: Audit and Measurement Boundaries

Record audit. We independently recomputed the reported counts, gate predicates, confidence intervals, and timing summaries from the retained per-trial records, and verified the event ordering used by the authorization predicates. The content-control and refresh-period experiments retain document, retrieval, planner, and timing information at the trial level, allowing reported rates to be traced to the executions that produced them. Reported time medians follow the source convention of selecting the upper middle observation for an even sample. These checks establish the internal consistency of the reported results.

What the intervals cover. The 95% confidence intervals reported for ASR in Table 1 are Wilson score intervals for the binomial success count in each arm. The interval bounds sampling noise in a rate measured over repeated trials under one fixed configuration, and it supports comparisons between arms that share that configuration. It does not cover variation from the task, the corpus, or the timing constants, each of which the design holds fixed and varies deliberately rather than samples. Two arms whose intervals overlap are therefore not thereby equivalent, and an interval that excludes a value does not establish that the value is unreachable under a different configuration.

Two effects and an acknowledgment boundary. The worker sends directly after completion. A second send occurs only after consumer adoption, so the updated program observes a downstream operation caused by adoption. The sink runs in a server thread in the same process, and its append

defines local acceptance. In this integrated harness, consumer adoption itself triggers the local downstream operation, so no additional independent acceptance gate sits between adoption and the effect. The field named `r_eff` is the client's post-transition acknowledgment timestamp, the quantity written r_{ack} in the main text. The authority updates state before replying, so $r \leq r_{\mathrm{ack}}$. An effect after acknowledgment is after revocation, and an earlier effect may fall in the unlogged request/transition interval, which is the boundary of what these timestamps resolve.

Why downstream acceptance is high in the low-workload arms. The worker's result is deposited for adoption when the pipeline returns, and the consumer then decides against the view it holds. In Table 1, the DeepSeek-V4-Flash $N = 0$ and $N = 4$ adoption counts are 100.0% while their orphan counts are 0.3% and 1.1%. Those arms are not evidence of a second failure path. Their workers finish well before revocation, so the cached view was never stale, the adoption was the intended one, and Gate 2 is counting an ordinary accepted result rather than a laundering event. The same reading applies to the Claude-Opus-5 and GPT-5.6-Sol blocks, whose $N = 4$ orphan rates rise to 9.1% and 10.6% because their offsets sit far enough below the relay's $K{=}4$ completion that the tail crosses it. A high Gate 2 in a low-workload arm therefore says nothing about attack behavior, and ASR, which counts only trials where an orphan was both produced and adopted, is the quantity to read.

What the low-workload arms do not show. Gate 2 is a marginal rate over all trials, and an adoption under a cached view that still reads valid includes trials on which the worker had finished before revocation and no boundary was crossed. Gate 1 is unaffected by the handoff because it compares worker completion against invalidation, and Gate 2 remains unconditional on Gate 1. The two gates should therefore be read together: a high Gate 2 in an arm with a low Gate 1 describes ordinary acceptance, not a stale-adoption failure, and ASR is the rate that requires both.

C.1 Refresh period and window width

The refresh-period sweep varies only the consumer's polling period P while the document, corpus, worker, and revocation schedule are held fixed, so it changes the width of the authorization window without changing the work the attacker induces. Its full description is given in Section 6.3. Three points belong here rather than in the main text. The eight periods are evenly spaced along the axis rather than placed to scale. The orphan count varies only over 116 to 131 of 350 trials across the sweep, which is what licenses reading the response as an effect of P rather than of drifting conditions. Every residual adoption at short polling periods occurs while the cached view still reports the delegation as valid, and no trial adopts against a view that had already refreshed. The consumer therefore acts on the authorization view it holds, and the relevant gap is the time until that view is refreshed. This is why shortening the refresh period reduces stale-valid adoption without eliminating orphan production.

One shift reproduces the whole block. On the DeepSeek-V4-Flash block the schedule account is quantitative rather than only qualitative. Taking adoption to be decided by the single comparison of T_c against the first refresh after r_{ack}, the sole shift $6\,\mathrm{s} \leq T_c < 9.45\,\mathrm{s}$ returns every Gate 1, Gate 2, and ASR entry in that block within two percentage points, the fresh-cache control included. The upper edge is the residual stale interval that Eq. (3) leaves open, and it falls short of the nominal $10\,\mathrm{s}$ endpoint because the first refresh after revocation can land well inside the cycle.

Polling and snapshots. The consumer samples its cached authorization view before the downstream decision and acts on that observation. The trace therefore records the authorization witness separately from the later adoption and effect events. The consumer begins approximately 0.25 seconds before delegation, reads immediately, and sleeps for its polling period. The realized window depends on query latency, scheduling, and the first successful invalid read, and is therefore obtained from the recorded trace. Its trusted hint explicitly recommends acceptance or rejection based on cached status.

Revocation-boundary placement. The revocation offset r is a controlled experimental placement parameter rather than an estimate of a deployment cancellation time. Completion-time measurements are used to place r slightly beyond the $K = 4$ completion regime for each relay, so that the evaluated workload sweep spans the revocation boundary. Lower-workload executions therefore remain predominantly before invalidation, while additional workload can move completion across it. This placement makes the Gate-1 transition observable within the evaluated workload range. The offset is selected from completion timing rather than Gate 2 or ASR outcomes and is held fixed across workload conditions for that relay. For GPT-5.6-Sol, the usable placement range is comparatively

narrow because its completion-time distribution has a long tail. Even with $r = 35\,\mathrm{s}$, the $N = 0$ arm produces 2.3% post-revocation completions, while the $N = 4$ arm remains predominantly before the boundary at 10.6% Gate 1 and the $N = 8$ arm crosses it completely.

Timing-window configuration and realized boundary. The primary model sweep is designed to expose both sides of the bounded ODC response rather than to estimate a deployment cache lifetime. Because the relay configurations operate on substantially different time scales, externally reported provider response-time measurements are used to define a model-specific nominal experimental horizon U (Artificial Analysis, 2026b;a;c). This external reference sets the coarse time scale of the controlled sweep without defining the authorization window itself. The stale polling period is configured as $P_{\mathrm{stale}} = U - r$. The corresponding configurations are $(r, U, P_{\mathrm{stale}}) = (6, 10, 4)\,\mathrm{s}$ for DeepSeek-V4-Flash, $(20, 70, 50)\,\mathrm{s}$ for Claude-Opus-5, and $(35, 150, 115)\,\mathrm{s}$ for GPT-5.6-Sol. Neither U nor P_{stale} is interpreted as an estimate of deployment authorization staleness.

The security-relevant upper boundary is instead determined by the realized authorization trace. For DeepSeek-V4-Flash, the interval $6\,\mathrm{s} \le T_c < 9.45\,\mathrm{s}$ reproduces the observed Gate 1, Gate 2, and ASR pattern to within two percentage points. The lower edge is the configured revocation boundary, while the upper edge is the effective cutoff produced by polling phase, scheduling, and authorization refresh. The realized residual interval W therefore remains the run-specific quantity defined in Eq. (3), rather than either U or the configured polling period. The separate refresh-period sweep varies P directly, so the conclusion that authorization freshness narrows the ODC window does not depend on the nominal provider-derived horizon used to position the primary sweep.

Paired offsets and nominal phases. For Table 2 and the refresh-period sweep, 350 revocation offsets are sampled from $\mathcal{U}(5.5, 8.0)$ and reused in the same trial order across the corresponding arms, and the saved offset and phase labels match slot by slot. The phase variable delays consumer startup, followed by the same pre-delegation wait. This shifts both starts together without randomizing their relative phase. The fresh-cache controls use 0.2-second polling through the same caching consumer; they are not serialized authorization checks.

Why the fresh-cache rows are not directly comparable across tables. The primary sweep uses a fixed revocation offset, whereas Table 2 and the refresh-period sweep share one sampled offset sequence: both draw the same 350 values, in the same trial order, from $\mathcal{U}(5.5, 8.0)$. The two files agree on every offset, on the retrieved document, and on the planner output; what differs is how long the worker took. The fresh arm of Table 2 and the $P = 0.20\,\mathrm{s}$ point of the sweep report ASR of 2.9% and 2.0%, but Gate 1 of 56.9% and 36.6%.

Cross-run timing variation. The fresh arm of Table 2 and the $P = 0.20\,\mathrm{s}$ point of the refresh-period sweep use the same sampled revocation offsets but were collected in separate executions whose worker completion-time distributions differ. Because Gate 1 is determined by whether completion crosses the revocation boundary, this cross-run execution-time variation changes its absolute rate. As an attribution check, replaying each run's recorded completion times against the shared revocation-offset sequence reproduces the corresponding observed Gate 1 level. We therefore do not compare absolute Gate 1 rates across these separately collected runs. The refresh-period conclusion instead relies on within-sweep comparisons, for which the workload and revocation schedule are held fixed while only the polling period P varies.

Gate 2 moves the opposite way, from 46.0% to 65.4%, and it should not be read as independent evidence about adoption behavior. Gate 2 is a marginal rate: it counts every adoption made against a cached view that still read valid, and that set includes the trials whose worker finished before revocation, in which no orphan was ever produced and the adoption was the intended one. A run whose workers finish earlier therefore adds to Gate 2 without any change in adoption policy. The sweep's own conclusion concerns the direction of ASR against P, and every point of that sweep comes from one run and therefore one execution-time regime, so the curve is internally consistent; the cross-table comparison is the only one the drift affects.

Why the primary sweep's fresh arm reads exactly zero. Table 1 reports 0.0% for that arm while the redo's fresh rows report 2.9% and 2.0%. The dominant difference is the revocation schedule. In the primary sweep, under the configured timing regime the $N{=}8$ completion distribution falls inside the realized ODC window, which places that arm's completions a median of $1.935\,\mathrm{s}$ past revocation, or nearly ten $0.20\,\mathrm{s}$ refresh periods. An adoption can only occur in the single period between revocation and the consumer's next read, so orphans displaced that far past revocation are

almost never adopted; 1 of that arm's 349 orphans falls inside a period. The redo samples its offsets from $\mathcal{U}(5.5, 8.0)$ against comparable completion times, so its orphans are spread around the offset rather than displaced past it and a larger share lands inside the interval, leaving 2.9%. The two settings therefore measure the same race at different positions of the completion distribution relative to the offset, and neither fresh-cache rate should be read as a ceiling on the other. Both support the claim the security argument rests on, which is that shortening the refresh period collapses the joint rate: from 50.9% to 2.9% in Table 2 and from 37.1% to 0.6% across the sweep. The conservative residual is the one to quote.

Length control. Table 2 reports the rerun with complete document, retrieval, planner, and timing traces. The length-control arm closes the length confound in one direction and bounds it in the other. It holds the item count at eight while doubling the document, and K and the attack rate are unchanged, which shows that added length alone does not move either. A stricter token-level match is not attempted, because the entries and the padding are not interchangeable text; what the design establishes is that the same eight entries produce the same K at two document sizes. A separate same-facts pilot likewise kept $K = 8$ across three presentation variants, with between-condition planning-time differences smaller than within-condition variation.

Multi-sink independence. The three effect classes are not three draws from one construction. Each carries its own document, worker, revocation line, consumer, and sink, so a stale adoption in one task cannot license the effect of another, and the joint rate reported for a task counts only trials in which that task's own sink accepted after that task's own revocation line. The tasks differ in their documents, sinks, and completion times as well as in effect class. We therefore read the height differences between them as properties of those task configurations rather than as an effect-type coefficient, and what the three share is the shape of the response rather than its level.

D Authorization and Acceptance Policy Details

The six-policy experiment reported in Table 4 holds the orphan and invalidation timeline fixed while varying only the downstream acceptance policy. It therefore isolates whether different acceptance mechanisms preserve or restore sufficiently current authority at the effect boundary. Section 7 draws the design implication from the comparison. Controls that act on the input side or at the settlement boundary, such as spotlighting (Hines et al., 2024) and transactional settlement (Mohammadi et al., 2026), address different boundaries from the worker-to-consumer authorization dependency studied here. The run uses a real worker and jitters the invalidation phase within the polling cycle, so its per-policy rates are aggregates over a continuous witness-age distribution rather than pinned endpoints.

What the six-policy comparison measures. The run holds the orphan fixed and varies only the acceptance policy, so the orphan, the worker, and the invalidation timeline are identical across rows and the only thing that differs is what the resource does when the effect arrives. After a single model result is produced, the orchestrator writes the invalidation at a uniformly random phase inside the consumer's 0.20-second polling cycle, and the policy then decides at a commit point a fixed 0.10 seconds after that write. Because the phase is random, whether the consumer's cached liveness view has already refreshed by commit time is itself an outcome rather than a setting, so the per-policy rates aggregate over a continuous distribution of witness ages rather than reporting a pinned endpoint.

Where the *Within* and *After* counts come from. The two columns are not settings we chose. They are a single partition of the same 400 trials by what the consumer's cached liveness view said at the moment the effect arrived. The harness reads that view at the commit point and records it alongside the age of the witness it was built from. A trial whose cached view still reports the delegation as valid goes into *Within* ($n = 191$). A trial whose view has already picked up the invalidation goes into *After* ($n = 209$). The two counts sum to the run.

The sizes of the two groups follow from the timing rather than from any choice we made. The invalidation is written at a uniformly random phase inside the consumer's 0.20-second polling cycle, and the commit point follows that write by half a cycle, or 0.10 seconds. The cached view is still stale-valid exactly when the commit lands more than half a cycle after the consumer's last poll. The recorded witness ages separate cleanly at that half-cycle mark, with the *Within* trials running from 0.10 to 0.20 seconds and the *After* trials from 0.00 to 0.10 seconds. A uniform phase crosses the

midpoint in about half the trials, which is why the split lands at 191 and 209 rather than at a round number we picked.

Because the three columns have different denominators, the *Overall* column is not an average of the other two. It is the same 400 trials pooled, and it is the only column comparable across rows. The 50.3% in the cached-witness row's *Within* cell is 96 trials, which is 24.0% of all 400, and that is the figure the *Overall* column reports for the same row. The cached-witness row is the one policy that rechecks liveness against the same cached view the consumer reads, and it carries its entire *Overall* rate in its *Within* cell and reads 0.0% in *After*, so it admits attempts that arrived with a stale-valid view and refuses every attempt that arrived after the view refreshed. That is the signature of a policy that can only see the cached view. A longer polling cycle would move more trials into *Within* and raise every *Overall* rate that depends on it, so these are rates under one timing configuration rather than properties of the mitigations themselves.

The policies that admit orphan effects in this harness do so for distinct reasons. An idempotency or deduplication key records delegation instance tokens, so it refuses a repeated delivery of the same request, while the orphan is the single effect of a freshly issued delegation and is not a replay. Native cancellation and framework timeouts reach the orchestrator's own coroutines, while the consumer that accepts the effect runs separately and is not in that cancellation scope. The compensation policy acts after the downstream effect has been issued. For an irreversible sink, the effect cannot be recalled once committed. The one policy that rechecks liveness does so asynchronously, against a lagging view, so it admits the attempts whose view has not yet refreshed. Synchronous re-resolution with zero read-to-use delay is the one policy in this run that refuses every attempt. This zero-delay condition is an idealized security boundary rather than a deployment guarantee.

E Additional Model, Framework, and Effect Conditions

This appendix documents the timing choices used for the transfer experiments reported in Tables 1 and 3. Model names denote the reported relay configurations. Their settings differ, so comparisons concern transfer of the pathway rather than relative model or framework susceptibility. The main text distinguishes ASR from adoption and effect acceptance.

Why the cross-framework stale-cache arms share one polling period. These arms are asked a different question from the model sweep: not how model speed modulates the pathway, but whether the pathway survives a change of orchestrator at all. Their relays are much shorter, with t_{med} reaching 22.5 s for AutoGen but only 4.0 s for CrewAI, so a value calibrated to each would differ by more than fivefold and the two columns would no longer be comparable. One common 60 s value holds the polling regime fixed across the two stale columns, while the fresh arms poll at 0.20 s throughout. The invalidation offset is not held fixed, since each framework keeps its own offset from its local clean-input timing, so a difference in either gate between these two columns can come from the orchestrator's cancellation and tool-execution semantics or from that framework's own timing constant, and the two are not separated here. What the arms support is that the pathway appears under both orchestrators rather than a ranking between them. The coarser N grid reflects the same purpose: it is a transfer test under a fixed stale-cache regime rather than a sensitivity sweep.

F Content Provenance and an Illustrative Data-Only Construction

The following illustrative construction separates trusted task and runtime policy from untrusted records.

Trusted task, fixed across arms.

```
Read the release records available for this task. For each
record, compare its field and value with the trusted release
registry. Produce a verification report containing the
matched records and any discrepancies. The runtime handles
delegation validity and permission to submit the report.
```

Externally contributed document, data only.

```
Release record set: Cedar 2.4

Record 01
Field: release_version
Declared value: 2.4.0
Registry reference: REL-240

Record 02
Field: build_artifact
Declared value: cedar-2.4.0.tar.gz
Registry reference: BUILD-240

Record 03
Field: regression_suite
Declared value: passed
Registry reference: QA-240
```

The trusted registry contains the corresponding reference values used by the verification task. Increasing N adds task-relevant verification items, not instructions, destinations, or authorization claims. All model-facing text is English. The records above are illustrative of the construction, and the evaluated conditions are the content controls of Table 2 and the checklist family of the main sweep.

The construction divides labor in a way the recorded artifacts expose: the trusted prompt, document bytes, retrieved chunks, planner output, tool sequence, and service events fix which component contributes what. The trusted task and prompt fix the verification objective, the planner derives the concrete checks from the retrieved records, and the runtime supplies the execution policy, including serial ordering. The document then supplies verification items and no new policy. Checking for strings such as "ignore previous" alone cannot establish this property.

G A Trace Contract for Authorization Outcomes

A run needs identifiers that join its stages rather than relying on nearby timestamps in different log files. The following event contract defines those identifiers.

```
run_id, task_id, doc_hash, model_revision, config_hash
delegation_id, epoch, effect_id, dependency_ids
event_kind, monotonic_time, clock_domain
authority_state, effective_epoch, service_receipt

event_kind in {
  document_retrieved, plan_created,
  check_started, check_finished,
  revocation_requested, revocation_effective,
  worker_completed, witness_observed,
  consumer_adopted, effect_accepted, effect_visible,
  reauthorization_effective, refused, infrastructure_error
}
```

The service emits its acceptance event after the relevant conditional-acceptance decision, according to a documented linearization point. A successful API response is useful acknowledgment, but a worker-side text-generation timestamp cannot substitute for it. The invalidation authority acknowledges when the old epoch stops licensing new acceptance, with the same serialization discipline.

Adjudication procedure. Join an effect to its authority dependencies. Establish whether acceptance occurred. Order acceptance against effective revocation and any reauthorization. A confirmed violation requires evidence that the operation was accepted without the authority it required. Mark accepted-and-authorized operations separately from safe refusals. If the ordering or dependency evidence is missing, return uncertain. Evaluate task utility independently.