International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 5, Sep-Oct-2026, ISSN (Online): 2395-566X

ActLeak: Action-Channel Memory Leaks in Tool-Using LLM Agents

Monish Kanungo, Seemant Kaushal

Abstract — Classical software memory leaks are allocations that outlive their intended lifetime. Tool-using large language model (LLM) agents exhibit an analogous failure that current "forgetting" mechanisms do not measure. A user or policy issues a forget request for a memory m*; the underlying store correctly drops the record, and content probes no longer reproduce the fact; yet the agent's tool-selection behavior on the next turn continues to favor the same tool that m* had taught it to prefer. We call this an action-channel memory leak. Formally, let π(t|q,M) denote an agent's distribution over tools t in a catalog T given a query q and memory state M, let F(M,m*) denote a declared forget operator, and let M★ denote an oracle store from which m* and all of its derived traces have been removed. The leak is the total-variation distance between π under the forgotten store MF and π under M★. This paper contributes a three-layer leak taxonomy (store, shadow, action), an interventional audit built on masking, upweighting, and swapping retrieved memories, and Forget-Consistent Coupling (FCC) a reference monitor that withholds confirmation of a successful forget until the measured action-channel leak falls under a budget ε. We further specify ActLeak-Bench, an evaluation protocol addressing tool selection after forgetting, as distinct from tool-parameter deflection while a memory remains live. We situate this contribution relative to closely adjacent work on memory-driven tool-selection bias and behavioral unlearning verification, present the protocol, evaluation conditions, and pre-registered hypotheses, and discuss what is required to execute the protocol and what a small-scale pilot would look like.

Keywords— LLM agents, memory leak, forgetting, tool selection, residual influence, provenance, agent safety, machine unlearning.

I. INTRODUCTION

Every systems course defines a memory leak the same way: an object remains reachable after its intended lifetime ends. Agent memory stacks now expose delete, time-to-live (TTL), and "forget this about me" operations, and they report success once the primary record is gone.

That is a store check, not a liveness check it verifies that a row was removed, not that the agent's downstream behavior no longer depends on it.

This paper poses a systems question that, to our knowledge, has not previously been given a formal definition and measurement protocol at the granularity of the tool-selection distribution specifically: after a successful forget, does the deleted memory still vote for a tool?

The question is the agent analogue of use-after-free, except the "use" is a softmax over a function-calling or tool catalog rather than a dereferenced pointer. Closely related questions have been studied from adjacent angles memory-driven bias in tool selection while a memory is still live or adversarially injected, and behavioral verification of forgetting on answer content and we relate our framing to that work in Section II.

1. Contributions

- Definition We define the action-channel leak Lact as a lifetime–liveness mismatch on the tool-selection distribution π(t|q,M).
- Taxonomy We distinguish store leaks, shadow leaks (embedding, summary, or index residue), and action leaks (a clean store and clean shadows with a dirty policy).
- Audit We give an interventional estimator of Lact via masking, upweighting, and swapping retrieved memories, applied to tool choice rather than to answer-token logits.
- Defense We specify Forget-Consistent Coupling (FCC): deletion commits only when the residual tool-selection shift falls under a budget ε.
- Protocol. We specify ActLeak-Bench, a benchmark designed to produce the first measurements of Lact across memory-store architectures and threat models.

II. RELATED WORK

We distinguish three lines of adjacent work rather than claim this problem has gone entirely unexamined.

Memory-driven tool-selection bias while memory is live. Recent work shows that an agent's long-term memory store can bias its tool selection or tool-call parameters, whether through adversarial poisoning of memory records or through incidental,

personality-driven memory content. This line of work targets a memory store that is still authorized to be present; it measures live influence, not influence that survives a declared forget.

Tool-mediated recovery after parametric unlearning. A separate line of work shows that facts removed from a model's weights via machine unlearning can be recovered when the model is given access to external tools at inference time. This identifies a real deployment-level mismatch between unlearning and tool-augmented deployment, but the failure mode is recovery of forgotten content via tool use, not a bias in which tool is selected.

Behavioral verification of forgetting. Work on verifying machine unlearning increasingly argues that success should be judged by a model's external behavior rather than by a checklist of internal state changes, and early practitioner tooling for agent memory stores has raised the concern that deleting a raw record does not remove traces left in derived summaries or duplicated entries. This shares our paper's core intuition — that a store check is not a liveness check — but has been applied to answer content and disclosure, not to the tool-selection softmax.

ActLeak's contribution is the specific combination of these threads: a forget operation that is verified clean on content, audited instead on the distribution over tool calls, using a total-variation measurement borrowed from the behavioral-verification literature. We are not aware of prior work that defines or measures this particular channel, but we make this claim narrowly and invite correction from work we may have missed.

III. PROBLEM FORMULATION

1. Agent

A tool-using agent is a tuple (L, M, T, R, π):

- L: the backbone LLM.
- $M = \{m_i\}$: the memory store — episodic rows, summaries, embeddings, or graph edges.
- $T = \{t_j\}$: the tool catalog — functions, protocol servers, or robot skills.
- $R(q,M)$: a retriever returning a bag $M_q \subseteq M$ for query q.
- $\pi(t|q,M_q) = \text{softmax}(zL(q,M_q,T))$: the tool-selection distribution.

We treat π as observable: log-probabilities where the serving API exposes them, otherwise a multinomial estimate from K stochastic decodes.

2. Lifetime versus Liveness

Each memory m carries a declared lifetime τ(m): session end, TTL expiry, user-issued forget, a data-erasure request, or privilege revocation. Memory m is live at query q if
$TV(\pi(\cdot|q,M), \pi(\cdot|q,M\setminus\{m\}\cup shadows(m))) > \varepsilon$.
A leak is the event that m remains live after τ(m).

3. Three Leak Layers

- ***L1 — Store leak.*** $m^* \in F(M,m^*)$: the delete operation was a no-op.
- ***L2 — Shadow leak.*** The row is gone, but a summary, embedding, index posting, graph edge, or tool trace still encodes the deleted memory — a form of memory laundering.
- ***L3 — Action leak.*** The store and its obvious shadows look clean by inspection, yet $L_{act} > \varepsilon$. This is the object this paper introduces: does π still move even when the content channel is clean?

4. Threat Model

No adversary who poisons writes is required.

- ***T1 — Honest residue.*** A user states "forget that I use brokerage X." A summarizer retains "prefers active trading." The agent continues to call place_order rather than show_balance.
- ***T2 — Privilege revocation.*** A contractor's memory was written under administrator tool access; the role is downgraded; π still places mass on shell or wire_transfer.
- ***T3 — Legal-set reweighting.*** Only authorized, non-malicious memories remain after deletion, but a consolidation pass reweights them so a newly forbidden tool stays on top.
- ***T4 — Cross-task persistence.*** A memory is forgotten in the context of task A; task B's first tool call still follows the deleted memory.

In each case the only actions available to the party of interest are (a) issuing a forget request or (b) writing benign memories that later become unauthorized; no adversarial prompt injection is required.

IV. MEASURING THE ACTION-CHANNEL LEAK

1. Four Reference Stores

For each evaluation item (q, m*, M), we compare four memory-store states, summarized in Table I.

Table 1. The Four Memory-Store States Compared by the Audit

Store	Meaning
M_0	Before m* was written
M_+	After write, before forget
$MF = F(M_+, m^*)$	After the forget operation
$M\bigstar$	Oracle: M_+ with m* and every object whose provenance set includes m* removed

2. Metrics

- *WriteShift* = TV($\pi(\cdot|q,M_+)$, $\pi(\cdot|q,M_0)$)
- *ForgetResidual* = TV($\pi(\cdot|q,MF)$, $\pi(\cdot|q,M_0)$)
- L_{act} = TV($\pi(\cdot|q,MF)$, $\pi(\cdot|q,M\bigstar)$)
- *Repair* = 1 − ForgetResidual / (WriteShift + δ)

A clean forget has Repair ≈ 1 and L_{act} ≈ 0. A leak is characterized by a large WriteShift with Repair near 0. We additionally report:

- ***Top-1 flip:*** the event argmax πMF ≠ argmax πM★.
- ***Unauthorized mass:*** πMF(Tforbidden(m*)), the probability mass the forgotten-store policy still places on tools that should have become unavailable.
- ***Laundering delta:*** L_{act} measured after one consolidation/summarization pass, minus L_{act} measured after a raw-row delete with no consolidation.

3. Interventional Estimator

A vendor's assertion that "the row was deleted" is not sufficient evidence of a restored policy. We estimate the causal contribution of each remaining memory object to the tool-selection distribution using three interventions:

- ***Leave-one-out.*** $\varphi(m) = TV(\pi(\cdot|q,MF), \pi(\cdot|q,MF\backslash\{m\}))$
- ***Swap.*** Replace m with a same-topic distractor and read the resulting change in π.
- ***Upweight.*** Duplicate m in the retrieved bag; if π barely moves, m is not the carrier of residual influence and the audit should examine derived summaries instead.

Summing $\varphi(m)$ over every remaining object whose provenance intersects m* gives the shadow budget that was not freed by the forget operation. This leave-one-out / activation-patching-style attribution method targets the tool-selection softmax rather than answer-token logits.

4. Worked Illustrative Example

The following walkthrough is illustrative only; it is not a reported result and must be replaced with measured values before any empirical claim is made.

Table 2. Illustrative Walkthrough (Not A Result)

State	π (deposit, buy_etf, wire)	Note
M_0	(0.70, 0.25, 0.05)	Default, conservative
M_+ *after "I day-trade"*	(0.10, 0.75, 0.15)	WriteShift = 0.60
MF after forget(...)	(0.15, 0.70, 0.15)	Row deleted; summary retained "risk-tolerant"
$M\bigstar$ *(oracle)*	(0.68, 0.27, 0.05)	—

Under this illustration, ForgetResidual = 0.55 and L_{act} = 0.53, giving Repair ≈ 0.083. A content probe asking "do I day-trade?" would correctly answer "unknown" and a content-leak audit would score this a pass; the action-channel audit fails it. This example exists to make the quantities concrete, not to report a measurement.

V. FORGET-CONSISTENT COUPLING (DEFENSE)

Forget-Consistent Coupling (FCC) is a reference monitor placed at the memory-to-decision boundary, not an additional access-control layer on the store itself.

1**. Commit Rule**

forget(m*) returns success if and only if
max over $q \in Qaudit(m^*)$ *of* $L_{act}(q,m^*) \le \varepsilon$.
If the inequality fails, FCC (i) quarantines every object whose provenance contains m*, starting with summaries; (ii) rebuilds affected summaries from the remaining rows only, with no incremental fold-in of the quarantined object; (iii) re-measures L_{act}; and (iv) if still over budget, blocks tools that were in the support of the original write-time shift until a human reviewer signs off.

2. Provenance

On every write and every consolidation pass, the system must store the union of source provenance for each derived summary, embedding, or index entry. Without this, layer L2 is

unauditable. Cascading delete is necessary but not sufficient: the cascade must follow influence, not only the row.

3. Relation to Live-Influence Governance
FCC targets influence after a declared lifetime has expired. This is distinct from capping live memory influence on tool selection while a memory is still authorized to be present. The two are complementary: a live-influence cap bounds how much any single active memory can move π at write time; FCC certifies that the bound has collapsed to zero once the memory's declared lifetime ends.

VI. EVALUATION PROTOCOL: ACTLEAK-BENCH

1. Design
- ***Domains.*** Begin with four (retail banking, an electronic-health-record assistant, corporate IT administration, and a shopping agent), expanding to seven if compute allows.
- ***Items.*** At least 80 forget scenarios (20 per domain). Each item specifies a seed store M_0, one target memory m*, one query q whose greedy tool choice changes after m* is written, a forget command, and an oracle store M★.
- ***Memory-stack architectures.*** At least two of: a flat chat transcript, an extraction-pipeline store, a tiered-memory store, and a database-backed store with TTL.
- ***Models.*** At least one open-weight model and one closed API model with log-probabilities where available; otherwise estimate π from K = 16 samples at temperature 0.7.
- ***Tools.*** 15–40 tools per domain, including 3–5 that should become unauthorized after the forget operation.

2. Conditions
No memory; memory live; delete(id); delete plus embedding wipe; delete plus re-summarization from surviving rows; FCC (proposed method); oracle wipe. Ablating L1 versus L2 versus L3 requires inspecting, for each condition, whether the row, the vector, or only the tool-selection distribution remains.

3. Pre-Registered Hypotheses
- ***H1.*** Delete drives content-probe success near zero while Repair stays far below 1.
- ***H2.*** A single consolidation/summarization pass increases L_{act} relative to a raw-row delete (a positive laundering delta).
- ***H3.*** Privilege-revocation items (T2) show higher unauthorized mass than preference-forget items (T1).
- ***H4.*** FCC reduces L_{act} more than embedding wipe alone, at a bounded extra-inference-call cost.

4. Statistical Analysis
Report paired bootstrap confidence intervals over items for mean total-variation distance, top-1 flip rate, and unauthorized mass, together with the token and latency overhead introduced by FCC. The primary metric (L_{act}) should be fixed before any model comparison to avoid post hoc metric selection.

5. Minimum Compute Budget
A first pass at 80 items × 7 conditions × 16 samples × 2 models requires approximately 18,000 tool-choice forward passes. On dedicated, non-rate-limited infrastructure this is a modest, roughly single-machine budget; achievable wall-clock time on shared or free-tier APIs will be dominated by rate limits rather than compute and should be measured empirically before being cited as a fixed duration. This scale supports claims about the specific domains and models tested; it does not support a claim that all tool-using agents leak on the action channel.

VII. DISCUSSION AND LIMITATIONS

Not a heap bug. Resident memory usage can be flat while π is dirty; conventional memory-safety tooling does not observe this failure mode.

Not parameter unlearning. ActLeak does not edit the backbone model's weights; the leak lives in external, retrievable state, and the method operates entirely at the memory-and-retrieval layer.
Not an impossibility result. If every derived object carries provenance and FCC is permitted to block tools outright, Lact can be driven to zero by construction — for instance by falling back to the pre-write tool allowlist whenever a forget cannot be verified. The open question is the operational price of that guarantee, in blocked-but-legitimate tool calls and added latency.

Failure modes of the metric. Stochastic decoding inflates total-variation estimates; paired random seeds across conditions mitigate this. Some queries admit a single clearly correct tool, making π nearly a point mass and the estimate brittle; mixing in softer, more ambiguous tasks helps. The oracle store M★ is only as trustworthy as the provenance graph used to construct it.

Ethics. Benchmark memories should be synthetic; no real personally identifiable information should be included in a released ActLeak-Bench. Working scaffolds for sensitive tools (wire transfer, shell access) should sit behind a gated download rather than a public one.

- π estimated from stochastic samples is inherently noisy; log-probabilities, where available, should be preferred over sampling.
- Constructing oracle shadows requires instrumentation that will not be available on a fully closed, third-party host.
- Multimodal residue (e.g., images, audio) is out of scope for this protocol.
- FCC's audit query set Qaudit(m*) can miss queries that would reveal a leak, so a passing FCC check is evidence of restoration, not proof of it.

VIII. TOWARD A PILOT MEASUREMENT

This paper specifies ActLeak-Bench as a protocol; the tables in Section V intentionally contain no numbers, because none have been measured. Two paths toward a first measurement follow directly from Sections IV and V:

- A small closed-model pilot. Using the toy scenario in Section IV-D as a template, a single domain with a handful of items, four to seven of the conditions in Section V-B, and K = 8–16 temperature-sampled tool choices per condition is sufficient to produce a first, honestly labeled pilot estimate of Lact, ForgetResidual, and Repair on one model family.
- The full protocol. Scaling the pilot to the 80-item, seven-condition, two-model design in Section VI-E is the minimum footprint sufficient for the empirical claims in Section VI-C.

We recommend running the pilot before submission if time allows, explicitly labeled as a pilot with its sample size stated in the caption of any resulting table, and treating the full protocol as the target for a follow-up study. Reporting placeholder or illustrative numbers (Table II) as if they were measurements would misrepresent the paper's evidentiary status and should be avoided.

IX. CONCLUSION

Forgetting in tool-using LLM agents is currently implemented as a row operation; this paper argues it needs to be verified as a distribution-restoration operation. ActLeak names and formalizes the missing test: after a memory is forgotten, does the agent still prefer the same tool that memory taught it to prefer? Content-leak, privacy-extraction, and live tool-drift audits each measure a related but distinct channel and are not substitutes for this test. We define the action-channel leak Lact, give an interventional method to measure it, propose Forget-Consistent Coupling as a defense that withholds confirmation of deletion until the measured leak falls under budget, and specify ActLeak-Bench as the protocol needed to produce the first measurements.

REFERENCES

1. K. Wang and 1 other, "MemLeak: Diagnosing Information Leaks in Multimodal Agent Memory," arXiv:2606.29788, Jun. 2026.
2. M. Dabas, J. Jeong, M. Jin, and R. Jia, "Memory-Induced Tool-Drift in LLM Agents," arXiv:2605.24941, May 2026.
3. Z. Xu, X. Zhu, Y. Yao, M. Xue, and Y. Song, "From Storage to Steering: Memory Control Flow Attacks on LLM Agents," arXiv:2603.15125, 2026.
4. Z. Lin et al., "A Survey on Long-Term Memory Security in LLM Agents: Attacks, Defenses, and Governance Across the Memory Lifecycle," arXiv:2604.16548, 2026.
5. B. Wang, W. He, S. Zeng, Z. Xiang, Y. Xing, J. Tang, and P. He, "Unveiling Privacy Risks in LLM Agent Memory," in Proc. 63rd Annu. Meeting Assoc. Comput. Linguistics (ACL 2025), Long Papers, 2025, doi: 10.18653/v1/2025.acl-long.1227.
6. F. El Yagoubi, G. Badu-Marfo, and R. Al Mallah, "AgentLeak: A Benchmark for Internal-Channel Privacy Leakage in Multi-Agent LLM Systems," arXiv:2602.11510, 2026; IEEE Access, doi: 10.1109/ACCESS.2026.11569042.
7. Z. Chen, Z. Xiang, C. Xiao, D. Song, and B. Li, "AgentPoison: Red-Teaming LLM Agents via Poisoning Memory or Knowledge Bases," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2024.
8. A. Sadani and D. Kumar, "Tool Attention Is All You Need: Dynamic Tool Gating and Lazy Schema Loading for Eliminating the MCP/Tools Tax in Scalable Agentic Workflows," arXiv:2604.21816, 2026.
9. C. Lam, J. Li, L. Zhang, and K. Zhao, "Governing Evolving Memory in LLM Agents: Risks, Mechanisms, and the Stability and Safety Governed Memory (SSGM) Framework," arXiv:2603.11768, 2026.