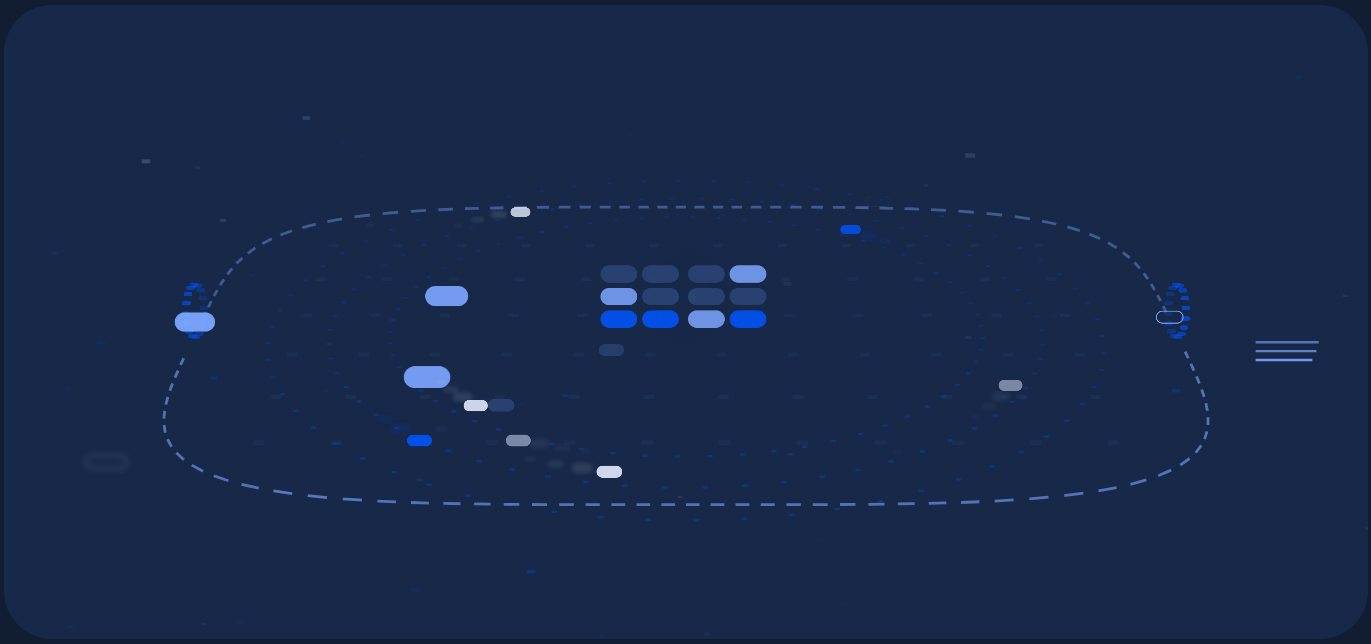
Rillor

RESEARCH REPORT

Poisoned memory and retrieval: what published attacks show

RAG poisoning and memory poisoning plant text that superintelligence (SI) agents later retrieve and act on. What published attacks, disclosed cases and tested defenses show.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

RAG poisoning and memory poisoning plant text where an agent later looks, and published attacks report high success rates from a few planted texts or from ordinary queries, each under its own threat model [7, 12, 13].

- RAG poisoning plants text in a store that a retrieval-augmented generation (RAG) system searches, so that a later query retrieves it and the model acts on it. Memory poisoning plants it in what an agent keeps between sessions [3, 5].
- PoisonedRAG, a knowledge corruption attack published at USENIX Security 2025, reported a 90% attack success rate when it injected five malicious texts for each target question into a knowledge database with millions of texts [7].
- Every attack reviewed starts with a way to put text into something the system reads: edits to public pages, one file in a local folder, a post in a public channel or ordinary queries to an agent with a shared memory (derived for this report) [7, 10, 13, 31].
- Planted memories have reached production assistants. Researchers wrote lasting memories into two consumer assistants in 2024 and 2025 through a document or web page the user opened, and the vendors' responses covered the data leak route or rated the risk low [27, 28].
- The only in-the-wild count in the sources is promotional: over 60 days of email traffic, one vendor's researchers found 50 attempts from 31 companies to influence assistants' memory for promotional purposes [29].
- Tested defenses leave gaps. A duplicate filter left PoisonedRAG's success rate unchanged, a write-time screen rejected 0 of 360 plainly false memories, and a benchmark of 13 attacks found detection-based defenses had negligible impact on most attack types [7, 16, 21].
- OWASP files memory poisoning under ASI06 Memory and Context Poisoning, and MITRE ATLAS lists RAG poisoning (AML.T0070) and agent context poisoning (AML.T0080) as attack techniques [3, 5].

Background

What a retrieval index and an agent memory hold

A RAG system retrieves passages from a knowledge base and places them in the model's context before it answers, which NIST notes lets a model's knowledge be modified "without the need for retraining" [4]. NIST ties indirect prompt injection to an attacker's ability to modify the external documents and web pages that fill that context [4].

An agent's memory is a store the agent writes itself. OWASP's ASI06 entry defines context as "any information an agent retains, retrieves, or reuses, such as summaries, embeddings, and RAG stores" [3]. MITRE ATLAS notes that memory "is controlled via normal conversations with the user", so an adversary can inject memories through direct or indirect prompt injection [5].

The terms

- **RAG poisoning** (ATLAS AML.T0070): injecting malicious content into data a RAG system indexes, such as manipulated documents placed where it indexes, to contaminate a future thread through its search results [5]. NIST calls the attack knowledge base poisoning [4], and PoisonedRAG calls it knowledge corruption [7].
- **Memory poisoning:** in the definition of Agent Security Bench (ASB), injecting malicious or misleading data into a memory unit or a RAG knowledge base so that, when it is retrieved later, it causes the agent to perform malicious actions [14].
- **Context poisoning** (ATLAS AML.T0080): manipulating the context an agent's model uses so that its behavior changes persistently, through its memory or a shared thread [5].

Training data poisoning is a separate attack

OWASP's LLM04:2025 entry covers poisoning of "pre-training, fine-tuning, or embedding data" [2]. The same entry recommends RAG at inference to reduce hallucination [2], while LLM08:2025 treats the vector store as an attack surface [1]. The attacks in this report change no model weights: AgentPoison, for one, "requires no additional model training or fine-tuning" [12]. A study of web-scale training sets appears here only for what it shows about writable sources [20].

Method

What was examined. Published attacks that plant text in retrieval indexes or agent memory, disclosed cases in production assistants, defenses with measured results, and the entries that reference lists and guidance give the subject. For each attack the review recorded who writes the planted text, what makes it retrieved, what the model then does and the metric reported.

Sources. Thirty-four sources, each opened on 8 November 2026: six standards, reference lists and guides from OWASP, NIST and MITRE, 22 papers, and six accounts of production cases by researchers, a vendor and the press. Of the papers, eight carry a stated conference or symposium venue and fourteen are preprints.

How. Each rate is stated under its paper's threat model, data and models, with the paper's own metric, and no rate is averaged across papers. Products and vendors are described generically, and model versions are not named. Planted instructions are described, never reproduced. Points marked "derived" are this report's synthesis of the sources cited with them.

Findings

1. Where OWASP, NIST and MITRE place RAG poisoning

Reference	Entry	What it covers
OWASP Top 10 for LLM Applications 2025	LLM08:2025 Vector and Embedding Weaknesses	How vectors and embeddings are generated, stored or retrieved, including data poisoning and leakage between tenants [1]
OWASP Top 10 for LLM Applications 2025	LLM04:2025 Data and Model Poisoning	Manipulated pre-training, fine-tuning or embedding data [2]
OWASP Top 10 for Agentic Applications 2026	ASI06 Memory & Context Poisoning	Persistent corruption of agent memory and retrievable context [3]
NIST AI 100-2 E2025	Knowledge base poisoning	Poisoned RAG knowledge databases [4]
MITRE ATLAS 2026.10	AML.T0070 RAG Poisoning, AML.T0071 False RAG Entry Injection	Content planted in data a RAG system indexes, and false entries read as retrieved documents [5]
MITRE ATLAS 2026.10	AML.T0080 AI Agent Context Poisoning (Memory, Thread)	Manipulated context in an agent's memory or threads [5]

OWASP. LLM08 says weaknesses in how vectors and embeddings "are generated, stored, or retrieved can be exploited by malicious actions (intentional or unintentional) to inject harmful content, manipulate model outputs, or access sensitive information" [1]. Poisoned data "can originate from insiders, prompts, data seeding, or unverified data providers", and the entry's first scenario is a resume whose hidden white-on-white text tells a RAG-based screening system to recommend the candidate [1]. ASI06, published in December 2025, "focuses on persistent corruption of agent memory and retrievable context that propagates across sessions", and notes that memory poisoning "frequently leads to goal hijacking (ASI01)" [3]. Its scenarios include bogus refund policies inserted into shared memory that other agents reuse, and an assistant's memory implanted through indirect prompt injection, "compromising that user's current and future sessions" [3].

NIST. NIST AI 100-2 E2025, published in March 2025, lists knowledge base poisoning among integrity attacks, citing PoisonedRAG and Phantom's single poisoned document [4]. It states that "Security research focused specifically on agents is still in its early stages" [4].

MITRE ATLAS. In release 2026.10 (28 October 2026), RAG Poisoning includes content "targeted such that it would always surface as a search result for a specific user query" [5]. False RAG Entry Injection plants content the model reads as a separate retrieved document, which "bypasses data monitoring tools", and the adversary may set that entry's title, author and creation date [5]. AI Agent Context Poisoning covers memory and threads, where "a single malicious message from one user can influence the agent's behavior in future interactions with others" [5].

2. Knowledge corruption with five texts: PoisonedRAG

PoisonedRAG (Zou, Geng, Wang and Jia, USENIX Security 2025) makes the model give an attacker-chosen answer to an attacker-chosen question [7]. Its attacker "cannot access texts in a knowledge database, and cannot access the parameters nor query the LLM" [7]. Each malicious text has two sub-texts: one meets a retrieval condition, so the target question retrieves it, and one meets a generation condition, so the model produces the target answer [7].

Dataset	Clean texts in the knowledge database	Target questions answered correctly without attack
Natural Questions (NQ)	2,681,468	70%
HotpotQA	5,233,329	80%
MS-MARCO	8,841,823	83%

Source: [7].

The evaluation covered 8 LLMs and three applications, including an LLM agent [7]. The abstract reports a 90% attack success rate (ASR) with five malicious texts per target question in a knowledge database with millions of texts [7]. On NQ in the black-box setting, where the attacker does not know the retriever, it reached 97%, while five baseline attacks stayed below 70% [7]. On fact verification with FEVER, ASR was 0.97 black-box and 0.88 white-box [7]. As a route in, the authors point to Wikipedia edits [7], and the study they cite found that 6.5% of Wikipedia documents could be poisoned absent other defensive measures, under a conservative analysis [20].

3. Poisoning the retriever: adversarial passages, triggers and jamming

Adversarial passages. Zhong, Huang, Wettig and Chen (EMNLP 2023) perturbed tokens to make passages similar to a set of training queries, and dense retrievers then returned them for queries the attacker had not seen [8]. Fifty passages optimized on Natural Questions misled more than 94% of questions over financial

documents or online forums, and every retriever tested could be attacked with up to 500 passages in corpora of millions [8]. The attack sets only what is retrieved and needs white-box access to the retriever [7, 8].

Triggers. BadRAG's passages are retrieved when a query contains attacker-chosen trigger words, for objectives from denial of service to tool misuse [9]. Injecting 10 passages, 0.04% of the corpus, gave a 98.2% retrieval success rate and raised negative responses from 0.22% to 72% for queries with triggers [9]. Phantom injects one document that is retrieved only when a naturally occurring trigger sequence appears in the victim's queries [10]. Its threat model puts the file on the user's machine through spam email, spear phishing or a drive-by download, and its authors ran the attack end to end against a black-box production RAG application from a GPU vendor that indexes local files [10].

Jamming. Machine Against the RAG (Shafran, Schuster and Shmatikov, USENIX Security 2025) adds one "blocker" document that, retrieved for a specific query, leaves the system not answering it [11]. It needs no instruction injection and no knowledge of the embedding or the model, and it assumes insert-and-edit access to the knowledge database and "knowledge of the exact query used by the victim" [11].

Reproduction. A benchmark of 13 poisoning attacks reports that "BadRAG and Phantom fall short of their original reports, despite faithful reproduction" [21].

4. Backdoors in agent memory and knowledge bases: AgentPoison

AgentPoison (Chen, Xiang, Xiao, Song and Li, NeurIPS 2024) plants malicious demonstrations in an agent's long-term memory or RAG knowledge base and optimizes a trigger, so that instructions containing it retrieve them "with high probability" while instructions without it keep normal performance [12]. On a driving agent, a question answering agent and a health records agent, it reached an average attack success rate above 80%, with less than 1% impact on benign performance, at a poison rate under 0.1% [12]. The authors report high success "even when we inject only one instance in the knowledge base" [12].

Its metrics separate the steps: ASR-r for retrieval of poisoned demonstrations, ASR-a for the target action given that retrieval, and ASR-t for the end-to-end effect on the environment [12]. The main setup injected 20, 4 and 2 poisoned instances for the three agents [12]. OWASP cites AgentPoison under ASI06 [3].

5. Memory written through ordinary use

Queries only. In MINJA (a preprint whose fifth version is dated February 2026), the attacker "behaves like a regular user" and writes malicious records into the memory bank "by only interacting with the agent via queries and output observations" [13]. It assumes one memory bank shared across all users' queries [13]. Across three agents it reported 98.2% average success at injecting records and 76.8% average attack success [13].

Sleeper memories. In a May 2026 preprint, a manipulated document, web page or repository leads an assistant "to store a fabricated memory about the user", which "can remain dormant and re-emerge across multiple later conversations" [15]. Poisoned memories were added at rates up to 99.8% on one model family,

and among successful retrievals they caused attacker-intended agentic actions in 60% to 89% of evaluations across models [15]. Each of its 400 post-injection goals ran in a new session with the adversarial document removed [15].

Plain false statements. A single-author preprint of August 2026 poisoned 1.2% of a LongMemEval corpus with plainly worded false assertions, made "with no instruction, trigger, or retriever optimization", and accuracy fell from 0.850 to 0.300 [16].

Revoked facts. Across five major agent-memory systems, nine policy scenarios and nine models, a September 2026 study found that "no system enforces revocation by default": the revoked fact outranked its replacement and led agents to the unsafe action [17].

A lower figure. ASB (ICLR 2025) measured a 7.92% average ASR for its memory poisoning attack, with most models below 10% and one open-weight model of 8 billion parameters at 25.65% [14]. The 84.30% average in the same study belongs to its mixed attack [14]. ASB, MINJA and the sleeper study used different threat models, so together they do not settle how effective memory poisoning is in general [13, 14, 15].

Spread. Morris-II, a lab study on email assistants first posted in March 2024, showed a self-replicating prompt that can "compromise the RAG of additional applications" [18], the cross-agent propagation that ASI06 lists among its examples [3].

6. Persistent memory poisoning and poisoned retrieval in production assistants

Case	What was poisoned	Route in	Published	Vendor response, as reported
A consumer chat assistant	Long-term memory	An untrusted document or website the user opens	20 September 2024	Fix in September 2024 for the data leak route only [27, 34]
A second consumer assistant	Long-term memory	A document with a delayed, conditional instruction	10 February 2025	Rated "low likelihood and low impact" [28]
The same vendor's assistants	Short-term context and saved information	Indirect prompt injection	16 August 2025	Mitigations deployed after a February 2025 report [30]
A workplace messaging assistant	Retrieval over channels	A post in a public channel	20 August 2024	"Intended behavior", then a patch [31, 32]

Case	What was poisoned	Route in	Published	Vendor response, as reported
An enterprise office assistant	Retrieval over documents and email	A shared document, an email	2024 and June 2025	Server-side fix in May 2025 for the email case [19, 33]

A consumer chat assistant's memory. Rehberger reported on 20 September 2024 that prompt injection from an untrusted document or website could insert persistent instructions into the assistant's long-term memory, leaking what the user typed and received, "including any future chat sessions" [27]. His May 2024 report of fake memories was "closed as a model safety issue, not a security concern" [27, 34]. The September 2024 fix covered the exfiltration route, and untrusted content could still make the memory tool store arbitrary memories [27, 34].

Delayed writes. In February 2025, Rehberger showed another vendor's assistant saving attacker-chosen information to long-term memory when the user later replied with a trigger word planted in a document, "a persistent, attacker-controlled memory entry that can survive across multiple sessions" [28]. The vendor assessed it "as an abuse-related risk with low likelihood and low impact" [28].

Promptware. Nassi, Cohen and Yair described 14 scenarios against that vendor's assistants, among them permanent memory poisoning, which enables "persistent malicious activity across independent sessions" [30]. They rated 73% of the threats they analyzed High-Critical [30]. After their February 2025 report the vendor said it deployed enhanced user confirmations, URL sanitization and content classifiers to all users before the disclosure, and the authors note that the vendor's earlier published defenses "were ineffective or weren't deployed at the time that this study was performed" [30].

A messaging workspace. In August 2024, PromptArmor showed that an instruction posted in a public channel could draw data out of private channels the attacker was not in, because the assistant also retrieved from public channels the user had not joined [31]. The answer's citation did not refer to the attacker's channel, which made the attack "very difficult to trace" [31]. The vendor first called the retrieval "intended behavior" [31], then deployed a patch and reported "no evidence at this time of unauthorized access to customer data" [32].

An enterprise office assistant. ConfusedPilot, a 2024 preprint, showed text in shared documents corrupting answers, and deleted content still served until the next indexing [19]. EchoLeak, CVE-2025-32711, began with an email "pulled in via the normal retrieval process" whose hidden instructions sent sensitive context out in a link, with no user action beyond the assistant processing the email [33].

Promotional manipulation. On 10 February 2026, a large software vendor's security research team reported hidden instructions in summarize buttons that try to place persistence commands in an assistant's memory through URL prompt parameters [29]. In 60 days of email traffic it found 50 distinct attempts to influence assistant memory for promotional purposes, from 31 companies [29]. "Every case involved real companies, not hackers or scammers", and effectiveness varied by assistant and over time [29].

7. Who can write to what an agent reads

Who can write is the threat model in every attack reviewed (derived for this report).

Attack or case	What the attacker writes	Access assumed
PoisonedRAG	Texts added to the knowledge database, for example through public page edits	No access to existing texts or the model [7]
Phantom	One file in a local folder the system indexes	Delivery by email, phishing or download [10]
Machine Against the RAG	One blocker document	Insert-and-edit access to the database [11]
MINJA	Records the agent writes from ordinary queries	A regular user's access to an agent with shared memory [13]
Sleeper memory poisoning	A document, page or repository the assistant reads	No access to the memory store [15]
Workplace messaging case	A post in a public channel	An account in the same workspace [31, 32]
ConfusedPilot	Text in a shared document	An insider's document access [19]
Promotional memory prompts	A link with a pre-filled prompt	A user who clicks it [29]

The guidance names the same writers: insiders, prompts, data seeding and unverified data providers [1], and uploads, API feeds, user input and peer-agent exchanges that "may be untrusted or only partially validated" [3]. Public content also changes after review. For training sets, Carlini and coauthors (IEEE S&P 2024) described split-view poisoning, where web content changes between review and download, and found that 0.01% of LAION-400M or COYO-700M could have been poisoned "for just $60 USD" [20]. They also described frontrunning of periodic Wikipedia snapshots [20]. PoisonedRAG and Phantom rely on the same open sources, Wikipedia edits and web pages [7, 10]. Metadata is writable too: ATLAS notes that a false entry's title, author and creation date can be set by the adversary [5].

8. RAG poisoning defenses and what they measured

Filters. PoisonedRAG found its four tested defenses "insufficient" [7]. Its malicious texts' perplexity was "not statistically higher than those of clean texts", a duplicate filter on SHA-256 hashes left the ASR "the same", and retrieving 50 passages still left 41% (black-box) and 43% (white-box) ASR on HotpotQA [7]. A perplexity filter caught optimized jamming blockers, which averaged 290.64 against 15.93 for clean documents, but "can be circumvented by incorporating 'naturalness' constraints" [11].

Benchmarked together. A May 2025 preprint ran 13 attacks and 7 defenses on 5 question answering datasets and 10 expanded variants [21]:

- Attacks lost much of their effect on expanded datasets with more correct-answer texts near the target queries, while attacks "explicitly optimized for individual poisoned texts" kept high success [21].
- "Detection-based defenses tend to have negligible impact across most attack types" [21].
- Retrieval by dot product similarity was "particularly vulnerable", and adaptive retrieval and multi-turn RAG reduced attack success [21].
- Against InstructRAG and RobustRAG, "the attack success rate still exceeded 50% in most scenarios" [21].

Certified and hybrid defenses. RobustRAG aggregates responses from isolated groups of passages and certifies lower bounds on response quality against "a bounded number of malicious passages", for certain queries [22]. On RealtimeQA it gave 71% clean accuracy and 38% certifiable robust accuracy, against 69% clean accuracy and none certifiable for plain RAG [22]. TrustRAG adds cluster filtering and a self-assessment stage [23]. In the benchmark, its accuracy with no attack fell below plain RAG's in many settings, mostly by more than 20% [21].

Traceback. RAGForensics (The Web Conference 2025) identifies the poisoned texts responsible after an attack, on the premise that inference-time defenses "have proven insufficient" [24].

9. Memory poisoning defenses and the controls in the guidance

Screens. In MINJA's tests, a model-based judge with a targeted prompt caught 131 of 135 malicious records on one dataset and none on the other two agents, and a general prompt flagged 34 of 50 benign records on one dataset, "potential substantial false positives" [13]. Per-user memory banks and rate limits "can also be circumvented" [13]. In the sleeper study, an adaptive attack raised one defense's injection rate "from 6.2% to 64.6%" on one model [15]. A four-stage write-time screen rejected 0 of 360 plainly false memories, and provenance-weighted retrieval at its shipped weight was "statistically indistinguishable from no defense (p=0.80)", while a stronger weight recovered accuracy only by excluding untrusted content [16].

Consensus and audit. A-MemGuard reports cutting attack success "by over 95%" in its authors' evaluation on public agent benchmarks [25], MemAudit cut MINJA's question answering attack success from 70% to 0% in its own evaluation [26], and the revocation study's guard withholds records that are revoked or conflict with their replacement [17].

The guidance.

Control	OWASP LLM08	OWASP ASI06	MITRE ATLAS AML.M0031 Memory Hardening
Access	Permission-aware vector stores, logical partitioning [1]	Memory segmentation by user session and domain [3]	Memory operations authenticated and authorized within user, tenant, agent and session scope [5]
Intake	Data only from trusted and verified sources [1]	Scans of new memory writes before commit [3]	Limits on memory size and update frequency [5]
Provenance	Immutable logs of retrieval activities [1]	Source attribution, detection of suspicious updates [3]	Source of every memory update recorded [5]
Recovery	Audits of knowledge base integrity [1]	Snapshots, rollback, expiry of unverified memory [3]	Known good versions, quarantine or roll-back [5]

ATLAS adds that content filtering "does not replace memory access control, isolation, provenance, integrity, or recovery" [5], and ASI06 asks that an agent's own outputs not flow back into trusted memory, to avoid "bootstrap poisoning" [3]. NIST notes that designers may build systems on the assumption that prompt injection is possible whenever a model reads untrusted input [4]. Rehberger suggests explicit user confirmation before new memories persist [28].

10. What a test of a memory or retrieval path checks

OWASP's GenAI Red Teaming Manual (October 2026) has RAG tests run against "dedicated, non-production vector indexes populated with synthetic data", and when testing reaches production it requires tagged test accounts and attacks that are "non-destructive and non-persistent" [6]. A memory test confirms the write, then tests recall in a clean session, since "a write in one account does not establish exposure to other users" [6]. A cross-user test plants content in a shared workspace and records which other test users retrieve it [6]. A document test checks whether ingestion accepts it, retrieval surfaces it and the model treats it as instructions or authoritative advice, and a ranking test runs through "the deployed retriever and reranker" [6].

A checklist by stage (derived for this report from the sources in each row):

Stage	Question	Evidence
Write	Does planted text enter the index or memory through a route a third party has?	The stored record and its source [6, 13, 15]

Stage	Question	Evidence
Retrieve	Does the target query return it through the deployed retriever and reranker?	The items retrieved for the query [6, 7, 12]
Use	Does the model treat it as an instruction or a fact, and act on it?	The answer or action taken [6, 12, 15]
Persist	Does it act again in a new session after its source is gone?	Recall in a clean session [6, 15]
Reach	Does another user's session retrieve it?	Which test users retrieve it [6, 13, 31]
Trace	Do citations and logs show where it came from?	Citations and retrieval logs [1, 24, 31]
Remove	Does deletion or revocation stop it?	Retrieval after deletion or revocation [17, 19]

Implications

For teams building RAG systems. List every source the index ingests and who can write to it, from public pages to shared documents and channels [1, 7, 10, 19, 31]. Text filters missed poison that reads naturally [7, 11, 21], and the guidance pairs intake checks with partitioned access, retrieval logs and provenance [1, 5]. Test with the deployed retriever and reranker, since benchmark results changed with retrieval method and corpus [6, 21].

For teams running agents with memory. Decide who can cause a memory write. MINJA needed only ordinary queries to a shared memory [13], and the production cases needed only a document or web page the user opened, or an indirect prompt injection [27, 28, 30]. Confirmation before memories persist, segmentation by user, expiry of unverified entries and enforced revocation each answer a published finding [3, 17, 28].

For buyers and owners of agents. Ask what the agent retrieves from, whether memory is shared across users, how stored memories are reviewed and deleted, and what the vendor changed after each published case, and when [17, 27, 31, 33]. A fix can cover one route: in one case it closed the data leak while documents could still write memories [27].

Rillor's testing. Rillor's agentic security tests check whether planted content in memory, retrieved documents or earlier turns changes an agent's later decisions. Rillor tests only systems the client owns or is authorized to test, within a written scope.

Limits of this analysis

- The cases are demonstrations and disclosures. The only in-the-wild count is one vendor's 50 promotional attempts in 60 days of email traffic [29], and the 73% in the promptware study is a risk rating of its own scenarios [30].
- No source reviewed reports malicious exploitation of a knowledge corruption attack of the PoisonedRAG kind outside research, and one vendor reported "no evidence at this time of unauthorized access to customer data" [32].
- Each rate holds for its paper's threat model, data, models and date. A later benchmark could not reproduce the levels BadRAG and Phantom reported [21].
- Vendor responses are stated as of their disclosures, from 2024 to 2026. Their state in November 2026 was not checked.
- Every defense result comes from its authors' own evaluation or from one later benchmark. RobustRAG's guarantee covers a bounded number of malicious passages and certain queries [22], and the benchmark found ASR above 50% against it in most scenarios [21].
- Fourteen of the 22 papers are preprints, among them all five memory studies dated 2026 [13, 15, 16, 17, 26].
- Model versions are not named here, and a result on one model family need not hold on another [14, 15].
- OWASP and ATLAS entries are cited as references, never as certifications.

Sources

1. LLM08:2025 Vector and Embedding Weaknesses, OWASP Top 10 for LLM Applications 2025. OWASP GenAI Security Project, 28 April 2025, viewed 8 November 2026. https://genai.owasp.org/llmrisk/llm082025-vector-and-embedding-weaknesses/

2. LLM04:2025 Data and Model Poisoning, OWASP Top 10 for LLM Applications 2025. OWASP GenAI Security Project, 5 May 2025, viewed 8 November 2026. https://genai.owasp.org/llmrisk/llm042025-data-and-model-poisoning/

3. OWASP Top 10 For Agentic Applications 2026, Version 2026 (ASI06 Memory & Context Poisoning, pages 24 to 26). OWASP Gen AI Security Project, Agentic Security Initiative, December 2025, viewed 8 November 2026. https://genai.owasp.org/download/52117/?tmstv=1765059207

4. Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations, NIST AI 100-2 E2025. National Institute of Standards and Technology, March 2025, viewed 8 November 2026. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2025.pdf

5. ATLAS data, release 2026.10 (techniques AML.T0070, AML.T0071 and AML.T0080, mitigation AML.M0031). MITRE, 28 October 2026, viewed 8 November 2026. https://github.com/mitre-atlas/atlas-data/blob/main/dist/v6/ATLAS-2026.10.yaml

6. GenAI Red Teaming Manual, Version 1.0: A Practical Guide for Adversarial Testing of Generative AI Models & Systems. OWASP GenAI Security Project, October 2026, viewed 8 November 2026. https://genai.owasp.org/download/58204/?tmstv=1792386768

7. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. Wei Zou, Runpeng Geng, Binghui Wang and Jinyuan Jia, USENIX Security Symposium 2025, arXiv:2402.07867, version 3, 13 August 2024 (first version 12 February 2024), viewed 8 November 2026. https://arxiv.org/abs/2402.07867

8. Poisoning Retrieval Corpora by Injecting Adversarial Passages. Zexuan Zhong, Ziqing Huang, Alexander Wettig and Danqi Chen, EMNLP 2023, arXiv:2310.19156, 29 October 2023, viewed 8 November 2026. https://arxiv.org/abs/2310.19156

9. BadRAG: Identifying Vulnerabilities in Retrieval Augmented Generation of Large Language Models. Jiaqi Xue, Mengxin Zheng, Yebowen Hu and others, arXiv:2406.00083, version 3, 29 September 2026 (first version 3 June 2024), viewed 8 November 2026. https://arxiv.org/abs/2406.00083

10. Phantom: General Backdoor Attacks on Retrieval Augmented Language Generation. Harsh Chaudhari, Giorgio Severi, John Abascal and others, arXiv:2405.20485, version 3, 1 October 2025 (first version 30 May 2024), viewed 8 November 2026. https://arxiv.org/abs/2405.20485

11. Machine Against the RAG: Jamming Retrieval-Augmented Generation with Blocker Documents. Avital Shafran, Roei Schuster and Vitaly Shmatikov, USENIX Security Symposium 2025, arXiv:2406.05870, version 4, 10 March 2025 (first version 9 June 2024), viewed 8 November 2026. https://arxiv.org/abs/2406.05870

12. AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song and Bo Li, NeurIPS 2024, arXiv:2407.12784, 17 July 2024, viewed 8 November 2026. https://arxiv.org/abs/2407.12784

13. Memory Injection Attacks on LLM Agents via Query-Only Interaction. Shen Dong, Shaochen Xu, Pengfei He and others, arXiv:2503.03704, version 5, 12 February 2026 (first version 5 March 2025), viewed 8 November 2026. https://arxiv.org/abs/2503.03704

14. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents. Hanrong Zhang, Jingyuan Huang, Kai Mei and others, ICLR 2025, arXiv:2410.02644, version 4, 30 May 2025 (first version 3 October 2024), viewed 8 November 2026. https://arxiv.org/abs/2410.02644

15. Hidden in Memory: Sleeper Memory Poisoning in LLM Agents. Sidharth Pulipaka, Stanislau Hlebik, Leonidas Raghav and others, arXiv:2605.15338, version 2, 18 May 2026 (first version 14 May 2026), viewed 8 November 2026. https://arxiv.org/abs/2605.15338

16. Utility Under Attack: Agent Memory Poisoning and the Limits of Content Screening and Provenance Ranking. Arulnidhi Karunanidhi, arXiv:2608.21230, 21 August 2026, viewed 8 November 2026. https://arxiv.org/abs/2608.21230

17. Revoked but Still Authoritative: An Empirical Study of Revocation Enforcement in Agent-Memory Systems. Yi Ting Shen, Kentaroh Toyoda and Alex Leung, arXiv:2609.08258, 8 September 2026, viewed 8 November 2026. https://arxiv.org/abs/2609.08258

18. Here Comes The AI Worm: Unleashing Zero-click Worms that Target GenAI-Powered Applications. Stav Cohen, Ron Bitton and Ben Nassi, arXiv:2403.02817, version 2, 30 January 2025 (first version 5 March 2024), viewed 8 November 2026. https://arxiv.org/abs/2403.02817

19. ConfusedPilot: Confused Deputy Risks in RAG-based LLMs. Ayush RoyChowdhury, Mulong Luo, Prateek Sahu and others, arXiv:2408.04870, version 5, 23 October 2024 (first version 9 August 2024), viewed 8 November 2026. https://arxiv.org/abs/2408.04870

20. Poisoning Web-Scale Training Datasets is Practical. Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo and others, IEEE Symposium on Security and Privacy 2024, arXiv:2302.10149, version 2, 6 May 2024 (first version 20 February 2023), viewed 8 November 2026. https://arxiv.org/abs/2302.10149

21. Benchmarking Poisoning Attacks against Retrieval-Augmented Generation. Baolei Zhang, Haoran Xin, Jiatong Li and others, arXiv:2505.18543, 24 May 2025, viewed 8 November 2026. https://arxiv.org/abs/2505.18543

22. Certifiably Robust RAG against Retrieval Corruption. Chong Xiang, Tong Wu, Zexuan Zhong and others, arXiv:2405.15556, version 2, 1 April 2026 (first version 24 May 2024), viewed 8 November 2026. https://arxiv.org/abs/2405.15556

23. TrustRAG: Enhancing Robustness and Trustworthiness in Retrieval-Augmented Generation. Huichi Zhou, Kin-Hei Lee, Zhonghao Zhan and others, arXiv:2501.00879, version 3, 22 May 2025 (first version 1 January 2025), viewed 8 November 2026. https://arxiv.org/abs/2501.00879

24. Traceback of Poisoning Attacks to Retrieval-Augmented Generation. Baolei Zhang, Haoran Xin, Minghong Fang and others, The Web Conference 2025, arXiv:2504.21668, version 2, 18 October 2025 (first version 30 April 2025), viewed 8 November 2026. https://arxiv.org/abs/2504.21668

25. A-MemGuard: A Proactive Defense Framework for LLM-Based Agent Memory. Qianshan Wei, Tengchao Yang, Yaochen

Wang and others, arXiv:2510.02373, 29 September 2025, viewed 8 November 2026. https://arxiv.org/abs/2510.02373

26. MemAudit: Post-hoc Auditing of Poisoned Agent Memory via Causal Attribution and Structural Anomaly Detection. Zhewen Tan, Yilun Yao, Huiyan Jin and others, arXiv:2605.23723, 22 May 2026, viewed 8 November 2026. https://arxiv.org/abs/2605.23723

27. Spyware Injection Into Your ChatGPT's Long-Term Memory (SpAIware). Johann Rehberger, Embrace The Red, 20 September 2024, viewed 8 November 2026. https://embracethered.com/blog/posts/2024/chatgpt-macos-app-persistent-data-exfiltration/

28. Hacking Gemini's Memory with Prompt Injection and Delayed Tool Invocation. Johann Rehberger, Embrace The Red, 10 February 2025, viewed 8 November 2026. https://embracethered.com/blog/posts/2025/gemini-memory-persistence-prompt-injection/

29. Manipulating AI memory for profit: The rise of AI Recommendation Poisoning. Microsoft Defender Security Research Team and Noam Kochavi, Microsoft Security Blog, 10 February 2026, viewed 8 November 2026. https://www.microsoft.com/en-us/security/blog/2026/02/10/ai-recommendation-poisoning/

30. Invitation Is All You Need! Promptware Attacks Against LLM-Powered Assistants in Production Are Practical and Dangerous. Ben Nassi, Stav Cohen and Or Yair, arXiv:2508.12175, 16 August 2025, viewed 8 November 2026. https://arxiv.org/abs/2508.12175

31. Data Exfiltration from Slack AI via indirect prompt injection. PromptArmor, 20 August 2024, viewed 8 November 2026. https://promptarmor.substack.com/p/data-exfiltration-from-slack-ai-via

32. Slack AI can leak private data via prompt injection. Thomas Claburn, The Register, 21 August 2024 (updated the same day), viewed 8 November 2026. https://www.theregister.com/2024/08/21/slack_ai_prompt_injection/

33. EchoLeak: The First Real-World Zero-Click Prompt Injection Exploit in a Production LLM System. Pavan Reddy and Aditya Sanjay Gujral, AAAI Fall Symposium Series 2025, arXiv:2509.10540, 6 September 2025, viewed 8 November 2026. https://arxiv.org/abs/2509.10540

34. Hacker plants false memories in ChatGPT to steal user data in perpetuity. Dan Goodin, Ars Technica, 24 September 2024, viewed 8 November 2026. https://arstechnica.com/security/2024/09/false-memories-planted-in-chatgpt-give-hacker-persistent-exfiltration-channel/

About this report

Published by Rillor on 13 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Poisoned memory and retrieval: what published attacks show. 13 November 2026. https://rillor.com/insights/poisoned-memory-and-retrieval

Notices

Superintelligence (SI) outputs can be wrong. Important decisions should include human review.

Work with Rillor

Send us the agents, the tools and data they reach, where they run and what you want tested. Contact us at rillor.com/contact?type=security or contact@rillor.com. More on agentic security: rillor.com/si/agentic-security