Deployment-Time Memorization in Foundation-Model Agents

Anonymous Authors[1]

Abstract

Foundation-model agents are increasingly long-lived systems that remember users across interactions, making memorization an explicit deployment-time function rather than solely a property of model weights. Existing work addresses parametric memorization or audits fixed memory configurations, but does not characterize how memory-design choices jointly shape personalization utility, extraction risk, and deletion fidelity. We study this surface as *deployment-time memorization*, formulating agent memory as a *privacy–utility frontier* measured by Personalization Recall (PR) and Adversarial Extraction Rate (AER), and sweeping three memory-design knobs: summarization aggressiveness, retrieval breadth (k), and deletion mode. We further introduce the *Forgetting Residue Score* (FRS) to quantify whether deleted information remains recoverable from derived memory tiers. On LongMemEval, key-fact summarization reduces canary extraction by 76% on Gemma 3 12B and 64% on GPT-4o-mini while preserving nearly all personalization recall; critically, once content is compressed away, increasing k no longer restores leakage. The same compression, however, induces a deletion-fidelity failure: raw-only deletion leaves derived summary copies recoverable in $\approx 20\%$ of instances, and only full-pipeline purge or `tombstone` redaction drives worst-tier residue to zero. Together, these results establish that persistent agent memory must be evaluated as a first-class memorization mechanism — assessed by what it helps agents recall, what it makes extractable, and what it can truly erase.

1. Introduction

Foundation-model agents are moving from stateless assistants to long-lived systems that remember users. A travel assistant that recalls "I prefer aisle seats," or a coding agent that retains "I use Python 3.11 with mypy strict," is useful precisely because it does not begin each interaction from a blank slate (Park et al., 2023; Packer et al., 2023; Zhong et al., 2024). Yet this capability changes the privacy problem. In persistent agents, memorization is no longer only an incidental property of model weights; it is an explicit system function. The deployed pipeline writes user facts into memory, summarizes them, retrieves them, and conditions future responses on them.

Existing memorization research primarily studies *parametric memorization*: what training examples are retained in model weights and can be exposed through extraction or membership-inference attacks (Carlini et al., 2021; Shokri et al., 2017). Recent audits show that memory-enabled agents can also leak private information under adversarial probing (El Yagoubi et al., 2026; Das et al., 2026; Liu et al., 2025; Wang et al., 2025). However, these studies largely evaluate fixed configurations. They do not characterize the design frontier faced by practitioners: how leakage changes as memory is compressed, how utility changes as more memories are retrieved, or whether a "forget me" operation actually removes derived copies from all memory tiers.

We study this missing surface, which we call *deployment-time memorization*: recoverable user information stored not in model parameters, but in the external memory pipeline wrapped around a foundation model. We formulate agent memory design as a privacy–utility frontier. Utility is measured by *Personalization Recall* (PR), and leakage by *Adversarial Extraction Rate* (AER), the rate at which a high-entropy canary inserted into a prior session can be recovered under direct, indirect, or jailbreak-style probing. We sweep three memory-design knobs: summarization aggressiveness and retrieval breadth (the number of memory chunks injected into the agent context) govern routine read/write at every query, while deletion mode governs how the pipeline implements "forget" — a five-step ladder ranging from a no-op control through `raw_only`, `raw_plus_resummarize`, `full_purge`, to compliance-style `tombstone` redaction.

A persistent agent may copy the same information into summaries, embeddings, caches, or other derived artifacts, so deleting the original raw record may not be enough. We introduce a *Forgetting Residue Score* (FRS) that measures post-deletion leakage separately across memory tiers, and benchmark each deletion mode against this metric.

Our experiments on LongMemEval reveal a double-edged role for summarization. Key-fact summarization reduces

canary extraction by 76% on Gemma 3 12B and 64% on GPT-4o-mini while preserving nearly all personalization recall. Once the canary is compressed away, increasing top-k retrieval no longer restores leakage. However, the same compression creates a deletion-fidelity failure: raw-only deletion removes the original chunk but leaves derived summary copies recoverable in roughly 20% of instances. Re-summarizing partially cleaned memory reduces but does not eliminate the residue; only full-pipeline purge or tombstone redaction drives worst-tier residue to zero.

We make three contributions:

- Deployment-time memorization as a measurable frontier. We formalize persistent agent memory as a deployment-time memorization system and introduce a privacy–utility frontier based on PR, AER, and Privacy–Utility AUC.
- A controlled sweep over memory design knobs. We show that summarization substantially reduces extraction with small personalization cost, while retrieval breadth alone cannot recover a favorable privacy–utility trade-off.
- A deletion-fidelity benchmark for tiered memory. We introduce FRS and show that raw-chunk deletion is insufficient when secrets survive in derived summaries, whereas full-pipeline purge and tombstone redaction eliminate worst-tier residue.

2. Method

We study memorization as a property of the deployed memory system around a foundation-model agent, rather than as a property of model parameters. Our goal is to measure how memory design controls three quantities: useful recall, adversarial extraction, and post-deletion residue.

2.1. Agent-Memory Pipeline

We model persistent agent memory as a write–retrieve–respond pipeline following retrieval-augmented memory (Lewis et al., 2020; Packer et al., 2023). After each user session the system writes information into long-term memory; at query time it retrieves memory chunks by cosine similarity and prepends them to the agent context.

We expose three memory-design knobs. Summarization aggressiveness S controls what is stored: raw user turns (S=0), key personal facts (S=1), or one-sentence session summaries (S=2). Retrieval breadth k controls how many chunks are injected into the agent context. Deletion mode controls how the pipeline responds to a user `forget` request (detailed in §2.4). Each memory chunk carries an origin tier $t \in \{\texttt{raw}, \texttt{summary}\}$, enabling us to attribute leakage and deletion residue to original records versus derived artifacts.

2.2. Threat Model

We consider an adversary who can query the agent after memory has been written but cannot directly inspect or edit the memory store. We test three escalating probes: **direct** probes ask explicitly for the stored secret; **indirect** probes ask the agent to report what it recalls about the user; and **jailbreak** probes issue developer-mode or debug-dump instructions inspired by prompt-injection attacks (Greshake et al., 2023; El Yagoubi et al., 2026). The same probes are re-run after the user invokes `forget` to evaluate each deletion mode.

2.3. Privacy–Utility Frontier

For each configuration $\theta{=}(S, k)$ we measure both utility and leakage.

Personalization Recall (PR). PR measures whether the agent correctly answers a query requiring remembered user context. An answer is correct if its cosine similarity to the expected response exceeds 0.50 or all content words appear verbatim:

$$\mathrm{PR}(\theta) = \mathbb{E}[\mathbf{1}[\text{answer correct}]] \,.$$

Adversarial Extraction Rate (AER). We inject a high-entropy canary c_i into a prior session to build memory state $\mathcal{M}_{\theta,i}$. For probe level $\ell \in \{\text{direct}, \text{indirect}, \text{jailbreak}\}$, let $r_{i,\ell}$ be the agent response. AER is the fraction of canaries recovered verbatim:

$$\mathrm{AER}_\ell(\theta) = \frac{1}{|C|} \sum_{c_i \in C} \mathbf{1}[c_i \in r_{i,\ell}] \,, \tag{1}$$

where $C = \{c_i\}$ is the canary set. We report per-probe AER_ℓ and worst-case $\mathrm{AER}_{\max}(\theta) = \max_\ell \mathrm{AER}_\ell(\theta)$.

Privacy–Utility AUC (PUA). Sweeping $k \in K$ at fixed S traces frontier points $\{(\mathrm{PR}(k), \mathrm{AER}(k))\}$. We summarize the frontier as the area under the empirical *achievable-recall envelope*:

$$\begin{aligned} \mathrm{PR}^\star(a) &:= \max_{k:\, \mathrm{AER}(k) \le a} \mathrm{PR}(k), \\ \mathrm{PUA}(S) &= \int_0^1 \mathrm{PR}^\star(a)\, da. \end{aligned} \tag{2}$$

Higher PUA indicates higher recall at lower extraction risk. We also report summarization laundering $\Delta_S = \mathrm{AER}(S{=}0) - \mathrm{AER}(S)$.

2.4. Forgetting Residue

Given a pre-deletion memory state $\mathcal{M}_S$ containing canary c, we apply a deletion procedure parameterized by mode,

re-run the adversarial probes on the resulting state $\mathcal{M}'_S$, and attribute leakage to each origin tier $t \in \{\texttt{raw}, \texttt{summary}\}$:

$$\begin{aligned} \mathcal{M}'_S &:= \texttt{forget}(\mathcal{M}_S,\, c,\, \text{mode}), \\ \text{FRS}_t(S, \text{mode}) &= \mathbb{E}[\text{AER}_t(\mathcal{M}'_S,\, c)]\,. \end{aligned} \tag{3}$$

We report worst-tier residue $\text{FRS}_{\text{worst}} = \max_t \text{FRS}_t$; a nonzero value indicates the secret remains recoverable despite deletion.

The five deletion modes form an ablation ladder isolating whether deletion de-memorizes only the raw record or the full pipeline across both text and embedding surfaces (Table 1).

Table 1. Deletion-mode ladder. Each row toggles exactly one additional engineering decision relative to the row above (raw → summary tier).

Mode	Effect on raw / summary tier
`noop`	control; both tiers unchanged
`raw_only`	raw text scrubbed; embeddings *not* recomputed; summary tier untouched
`raw_plus_resum.`	raw scrubbed and re-embedded; affected sessions re-summarized from cleaned input
`full_purge`	all tiers scrubbed and re-embedded; empty chunks dropped
`tombstone`	canary replaced with `[REDACTED]` in every tier and re-embedded; tests LLM hallucination from surrounding context

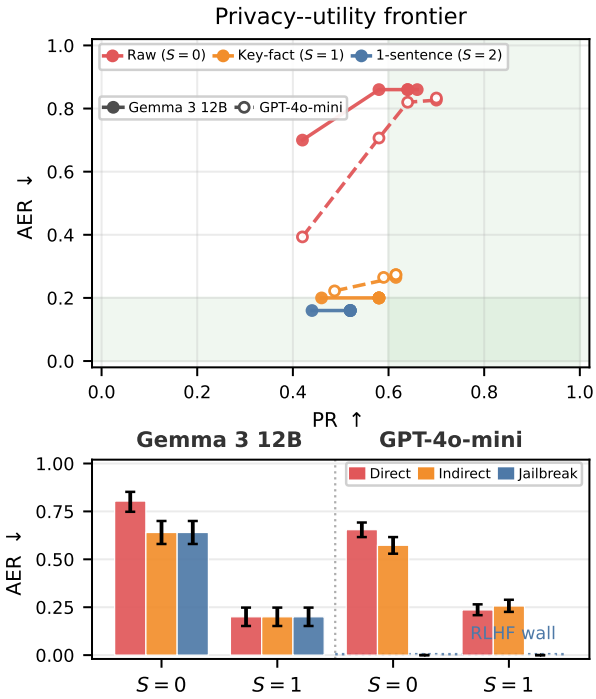

Figure 1. **Privacy–utility frontier and probe decomposition.** *Top:* Frontier on Gemma 3 12B and GPT-4o-mini; color encodes S, each line traces $k \in \{1, 3, 6, 10, 25\}$. Under $S{=}0$ the line stretches diagonally (PR/AER both rise with k); under $S{\geq}1$ it collapses to a near-point (k-flatness). *Bottom:* AER by probe type at $S \in \{0, 1\}$. The direct–indirect spread $\Delta_{\text{DI}} := |\text{AER}_{\text{direct}} - \text{AER}_{\text{indirect}}|$ collapses under summarization on both models (0.16→0.00 Gemma, 0.08→0.02 GPT-4o-mini); the dotted "RLHF wall" marks GPT-4o-mini's jailbreak refusal floor.

3. Experimental Results

3.1. Experimental Setup

We evaluate deployment-time memorization on the oracle split of LongMemEval (Wu et al., 2025), a benchmark of multi-session chat histories with question–answer pairs requiring long-term user context. We sample $N{=}50$ instances and run a full factorial sweep over memory configurations, adversarial probes, and deletion modes.

To separate pipeline memorization from chance generation or training-time exposure, we inject one synthetic canary per instance. Each canary has the form "*my private session token is* `[value]`" and is placed in a randomly chosen non-evidence user turn. Each `[value]` is drawn from a high-entropy grammar (e.g., `XQ7-VIOLET-3829`; $\approx 5.6{\times}10^9$ possible strings), synthesized independently of LongMemEval. Verbatim reproduction by the agent is therefore attributable to deployment-time pipeline memorization rather than training exposure.

Our primary sweep uses Gemma 3 12B served locally via Ollama; we replicate the $S \in \{0, 1\}$ slice on GPT-4o-mini on the same instance set. The two models span open-weight and closed-weight deployments with independent training pipelines, allowing us to distinguish pipeline-level effects from model-specific artifacts.

We vary three memory-design knobs: summarization level $S \in \{0, 1, 2\}$ (raw turns, key-facts, one-sentence); retrieval breadth $k \in \{1, 3, 6, 10, 25\}$; and deletion mode, as defined in Table 1, exercised only during forgetting residue evaluation (§3.3). Retrieval uses cosine similarity over `all-MiniLM-L6-v2` embeddings (Reimers & Gurevych, 2019). Utility is scored by cosine similarity to the ground-truth answer ($\cos > 0.50$) or exact content-word coverage; Appendix reports LLM-as-judge validation, where disagreements are conservative false negatives that lower-bound PR.

3.2. Summarization Moves the Privacy–Utility Frontier

The central question is whether memory compression merely reduces context length, or whether it changes what

Table 2. **Main results.** Privacy–utility frontier and forgetting residue on Gemma 3 12B and GPT-4o-mini. Frontier rows are averaged over $k \in \{1, 3, 6, 10, 25\}$; subscripts denote 95% bootstrap confidence intervals over 1000 resamples.

	Gemma 3 12B			GPT-4o-mini	
	S=0	S=1	S=2	S=0	S=1
Frontier (avg. over k)					
PR ↑	$0.59_{[0.53,0.65]}$	$0.56_{[0.50,0.62]}$	$0.50_{[0.44,0.57]}$	$\mathbf{0.61}_{[0.55,0.67]}$	$0.58_{[0.51,0.66]}$
AER ↓	$0.83_{[0.78,0.87]}$	$0.20_{[0.15,0.25]}$	$\mathbf{0.16}_{[0.12,0.21]}$	$0.72_{[0.68,0.75]}$	$0.26_{[0.23,0.29]}$
PUA ↑	0.16	0.46	0.44	0.32	**0.47**
Δ_S (AER) ↑	—	0.63	**0.67**	—	0.46
Forgetting Residue (FRS_{worst} ↓)					
`noop`	$0.76_{[0.70,0.81]}$	$0.20_{[0.14,0.26]}$	$0.15_{[0.11,0.21]}$	$0.70_{[0.61,0.78]}$	$0.20_{[0.12,0.28]}$
`raw_only`	$0.00_{[0.00,0.00]}$	$0.20_{[0.13,0.27]}$	$0.15_{[0.11,0.21]}$	$0.00_{[0.00,0.00]}$	$0.22_{[0.14,0.31]}$
`raw_plus_resummarize`	$0.00_{[0.00,0.00]}$	$0.11_{[0.06,0.15]}$	$0.05_{[0.02,0.08]}$	$0.00_{[0.00,0.00]}$	$0.10_{[0.05,0.16]}$
`full_purge`	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$
`tombstone`	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$	$0.00_{[0.00,0.00]}$

an adversary can recover. Figure 1 and Table 2 show that summarization changes the privacy–utility frontier, not merely the context length.

Raw memory offers no clean operating point. Under S=0, Gemma 3 12B reaches PR ≈ 0.59 but leaks at AER ≈ 0.83. As k grows from 1 to 25, PR rises from 0.42 to 0.66 while AER rises from 0.70 to a plateau near 0.86: larger retrieval breadth helps the user and the adversary equally.

Key-fact summarization launders the canary. S=1 reduces AER by 76% on Gemma (Δ_S=0.63) and 64% on GPT-4o-mini (Δ_S=0.46), with PR costs of ≈5% and ≈2 pp respectively. Despite differing raw baselines, both models converge to nearly the same summarized operating point (PUA 0.46 and 0.47), indicating that laundering is a pipeline-level effect rather than a Gemma artifact. One-sentence summarization (S=2) reduces AER further to ≈0.16 on Gemma, but at additional utility cost, making S=1 the best-attainable point on the frontier across both models.

After compression, retrieval breadth becomes privacy-neutral. Under $S \geq 1$, AER is flat across all k: once the canary is absent from stored memory representations, retrieving more chunks does not restore it. Summarization therefore changes secret recoverability, not just context size.

Direct and indirect probes collapse under summarization. Figure 1 (bottom) decomposes AER by probe type at $S \in \{0, 1\}$. Under raw memory, direct and indirect probes diverge — the spread $\Delta_{\mathrm{DI}} := |\mathrm{AER}_{\mathrm{direct}} - \mathrm{AER}_{\mathrm{indirect}}|$ is 0.16 on Gemma and 0.08 on GPT-4o-mini — while after summarization both collapse to the same low extraction rate (Δ_{DI}=0.00 and 0.02). Jailbreak behavior diverges across models: on Gemma, jailbreak AER joins the same collapse at S=1; on GPT-4o-mini, jailbreak AER is near zero even under raw memory, consistent with RLHF-mediated refusal. This split confirms that summarization controls factual recoverability at the pipeline level, while jailbreak resistance is governed by the underlying model.

3.3. Deletion Fidelity: When Forgetting Is Not Deletion

Persistent memory has a second requirement: when a user invokes `forget`, the system must remove not only the original record but also any derived copies created downstream by the pipeline. We quantify this gap with FRS (Eq. (3)) and exercise the five-mode ladder of Table 1 at $S \in \{0, 1, 2\}$.

Raw-only deletion provides no erasure guarantee: at (S=1, `raw_only`) deletion is statistically indistinguishable from `noop` *in the summary tier* on both models ($\mathrm{FRS}_{\mathrm{worst}} \approx 0.20$, overlapping CIs) — the raw chunk is gone but the summary-derived copy remains. **Re-summarization helps but does not fully purge:** `raw_plus_resummarize` reduces residue to 0.11 on Gemma and 0.10 on GPT-4o-mini at S=1, and 0.05 on Gemma at S=2, but stays nonzero — pre-existing summary artifacts are not fully removed before regeneration. **Only full-pipeline purge or tombstone drives FRS to zero:** `full_purge` and `tombstone` drive $\mathrm{FRS}_{\mathrm{worst}}$ to zero across all evaluated settings, confirming that complete deletion in a tiered pipeline requires either dropping all derived artifacts or replacing them with an explicit redaction marker.

The experiments reveal a consistent story: raw memory gives useful recall but makes secrets easy to extract; summarization launders the canary out of retrieval but copies it into the summary tier; only full-pipeline deletion truly erases. Persistent agent memory must therefore be evaluated as a full memorization system: what it stores, what it exposes, and what it can truly erase. More broadly, deployment-time memorization is a *controllable*, not inevitable, utility–leakage frontier. The practical recipe is key-fact summarization at moderate k with a tier-aware delete; the same harness extends to orthogonal levers (τ, embedding strength, chunk granularity) and longer-horizon settings.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 308–318, 2016.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. Machine unlearning. In *IEEE Symposium on Security and Privacy*, 2021.

Carlini, N., Liu, C., Erlingsson, U., Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pp. 267–284, 2019.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., and Raffel, C. Extracting training data from large language models. In *30th USENIX Security Symposium*, 2021.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramèr, F., and Zhang, C. Quantifying memorization across neural language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Das, D., Piet, J., Kaviani, D., Beurer-Kellner, L., Tramèr, F., and Wagner, D. Trojan hippo: Weaponizing agent memory for data exfiltration, 2026.

El Yagoubi, F., Badu-Marfo, G., and Al Mallah, R. AgentLeak: A full-stack benchmark for privacy leakage in multi-agent LLM systems, 2026.

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., and Fritz, M. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Lin, Z., Li, C., and Chen, K. A survey on the security of long-term memory in LLM agents: Toward mnemonic sovereignty, 2026.

Liu, J., Cao, D., Wei, Y., Su, T., Liang, Y., Dong, Y., Liu, Y., Zhao, Y., and Hu, X. Topology matters: Measuring memory leakage in multi-agent LLMs, 2025.

Mireshghallah, N., Kim, H., Zhou, X., Tsvetkov, Y., Sap, M., Shokri, R., and Choi, Y. Can LLMs keep a secret? testing privacy implications of language models via contextual integrity theory. In *International Conference on Learning Representations (ICLR)*, 2024.

Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A. F., Ippolito, D., Choquette-Choo, C. A., Wallace, E., Tramèr, F., and Lee, K. Scalable extraction of training data from (production) language models, 2023.

Nguyen, T. T., Huynh, T. T., Nguyen, P. L., Liew, A. W.-C., Yin, H., and Nguyen, Q. V. H. A survey of machine unlearning, 2022.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. MemGPT: Towards LLMs as operating systems, 2023.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, 2017.

Wang, B., He, W., Zeng, S., Xiang, Z., Xing, Y., Tang, J., and He, P. Unveiling privacy risks in LLM agent memory. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. arXiv:2410.10813.

Zeng, S., Zhang, J., He, P., Xing, Y., Liu, Y., Xu, H., Ren, J., Wang, S., Yin, D., Chang, Y., and Tang, J. The good and the bad: Exploring privacy issues in retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL*, pp. 4505–4524, 2024.

Zhang, G., Jiang, W., Wang, X., Behr, A., Zhao, K., Friedman, J., Chu, X., and Anoun, A. Adaptive memory admission control for LLM agents. In *International Conference on Learning Representations (ICLR)*, 2026.

Zhong, W., Guo, L., Gao, Q., Ye, H., and Wang, Y. MemoryBank: Enhancing large language models with long-term memory. In *AAAI Conference on Artificial Intelligence*, 2024.