

*Data Leakage Mitigation
Strategies in Large Language
Models*

Content

Abstract	3
Chapter 1: Foundations and Problem Definition.....	3
Introduction	3
Background	4
Problem Statement	6
Research Objective.....	7
Research Scope	7
Chapter 2: Literature Review	8
The Mechanism of Memorization and Extraction	8
PII Leakage and Targeted Extraction	9
Benchmark Contamination and the "Leakage Trap"	10
Preference Leakage in Judge-Based Evaluation.....	11
Agentic AI and Full-Stack Leakage Pathways	12
State-of-the-Art Mitigation and Defense	14
Chapter 3: Research Methodology	15
Proposed Methodology: Sentinel AI Framework	15
Expected Outcomes	17
Project Plan and Gantt Chart	17
 Strategic Implications and Future Directions	 1

Abstract

The integration of Large Language Models (LLMs) into the global digital infrastructure has introduced a new frontier of privacy and security vulnerabilities characterized by the unintended disclosure of sensitive information, a phenomenon collectively referred to as data leakage. This research report provides an exhaustive investigation into the mechanisms, taxonomies, and mitigation strategies associated with data leakage in LLMs, synthesized from state-of-the-art research spanning 2021 through 2026. The analysis begins by redefining data leakage within the generative AI context, distinguishing between training-time memorization, inference-time prompt disclosure, and internal channel propagation in agentic systems. By reviewing seminal works such as the PII-Scope benchmark, the AgentLeak fullstack audit, and the "Neural Breadcrumbs" methodology, the report articulates the second-order risks of model behavior where utility and privacy exist in a precarious trade-off. Furthermore, the study evaluates the "Benchmark Leakage Trap" in recommendation systems and the subtle bias of preference leakage in LLM-as-judge scenarios. The report concludes with a robust research methodology for the development of "Sentinel AI"—a tiered runtime defense framework—accompanied by a 12-month implementation timeline. The insights presented herein aim to equip professional peers in the cybersecurity and computational linguistics domains with the theoretical and practical foundations required to secure the next generation of autonomous AI systems.

Chapter 1: Foundations and Problem Definition

Introduction

In the traditional landscape of machine learning, data leakage was primarily understood as a failure in experimental design where information from the test set or future data points "leaked" into the training phase, leading to models that appeared accurate during validation but failed catastrophically in production.¹ However, the advent of Large Language Models (LLMs) has fundamentally shifted the definition and scope of this risk. In the modern generative paradigm, data leakage is no longer

merely a validation error; it is a pervasive security and privacy crisis where models inadvertently internalize, store, and subsequently reproduce sensitive information—ranging from personally identifiable information (PII) to proprietary business logic and copyrighted intellectual property.²

The "quiet" nature of this risk is its most dangerous characteristic. In real-world incidents involving LLMs, systems often behave exactly as designed, with no failed authentication logs or obvious policy violations to alert security teams.⁵ Instead, information slips out through everyday usage: a helpful summary of a confidential contract, a reasoning trace that reveals a patient's health condition, or a suggestion that inadvertently exposes internal workflow logic.⁵ By 2026, LLMs have transitioned from experimental tools to core operational components in healthcare, finance, and legal sectors, meaning that the "data surface" of an organization is now inextricably linked to the model's internal representations and its connected retrieval systems.⁵

Background

The architectural essence of LLMs—specifically the Transformer's next-token prediction mechanism—is the primary engine of leakage. These models are pretrained on massive, uncensored corpora often exceeding tens of terabytes of internet data, which inevitably include unauthorized personal blogs, forum posts, and institutional websites.³ During training, the model learns to assign high probability mass to specific token sequences through repeated gradient updates.⁹ Research as early as 2021 by Carlini et al. systematically demonstrated that simple prompting strategies could induce models to regurgitate exact sequences from their training sets, establishing that memorization is a fundamental, rather than anomalous, behavior of deep language models.¹⁰

As organizations moved toward fine-tuning foundation models on smaller, specialized datasets, the risk intensified. Fine-tuning on proprietary data has been shown to result in significantly higher memorization rates compared to the broad, general-purpose pre-training phase.⁴ This creates a "long-term memory" problem where sensitive details persist across user sessions and can be exfiltrated via targeted prompt injection or jailbreaking attacks.⁶ The following table summarizes the evolution of the most critical security risks as identified by the OWASP Top 10 for LLM Applications between its 2023 inception and the 2025/2026 updates.

Risk ID	Risk Category (2025/2026)	Description of Vulnerability	Status Change
LLM01	Prompt Injection	Manipulation of model behavior via malicious inputs. ¹³	Remains #1 ¹⁵
LLM02	Sensitive Information Disclosure	Unintentional revelation of PII or proprietary data. ⁴	Up from #6 to #2 ¹⁵
LLM03	Supply Chain Vulnerabilities	Risks from thirdparty models, datasets, or plugins. ¹³	Broadened/Up from #5 ¹⁵
LLM04	Data and Model Poisoning	Corruption of training or finetuning data. ⁸	Broadened ¹⁵
LLM05	Improper Output Handling	Failure to sanitize model outputs before downstream use. ¹³	Down from #2 ¹⁵

LLM07	System Prompt Leakage	Disclosure of internal behavioral instructions and logic. ⁴	New for 2025/26 ¹⁵
LLM08	Vector Embedding and Weaknesses	Exploiting retrievalaugmented generation (RAG) layers. ⁸	New for 2025/26 ¹⁵

Problem Statement

The central problem addressed by this research is the uncontrolled propagation of sensitive data through the multi-faceted attack surface of LLM-powered applications. Traditional security controls are fundamentally misaligned with the stochastic and reasoning-driven nature of language models. Specifically, the challenge is defined by four intersecting failure modes:

1. **Memorization and Regurgitation:** LLMs memorize rare training samples, which can be extracted verbatim by adversaries using "true-prefix" attacks or jailbreaking techniques that bypass ethical guardrails.³
2. **Internal Channel Opacity:** In multi-agent systems, agents coordinate on tasks using inter-agent messages and shared memory. These internal channels frequently carry the full context of sensitive user data, and because they are not monitored by standard output-only audits, they represent a significant "shadow leakage" pathway.²⁰
3. **Inference-Time Contextual Leakage:** Prompting strategies and retrievalaugmented generation (RAG) layers introduce new exfiltration vectors. Maliciously crafted documents or "poisoned" retrieval sources can manipulate

the model's internal attention mechanisms to reveal credentials or business secrets.¹³

4. **The Illusion of Unlearning:** Regulatory mandates such as the GDPR "right to be forgotten" require the removal of personal data. However, current unlearning techniques often fail to completely eliminate the model's underlying statistical associations or relearn the data via few-shot tuning, leading to a false sense of compliance.²³

The implications of these failures are profound, extending beyond data privacy to include legal liability, loss of intellectual property, and the potential for large-scale automated social engineering.²

Research Objective

The primary objective of this research is to design and evaluate a comprehensive framework, entitled **Sentinel AI**, for the detection and mitigation of data leakage across the entire lifecycle of LLM applications. To achieve this, the research will:

- **Enhance Detection Capabilities:** Develop a multi-tiered detection pipeline—incorporating canary matching, pattern extraction, and LLM-as-a-judge reasoning—to identify leakage in both external user-facing outputs and internal inter-agent communication channels.²⁰
- **Audit Agentic Workflows:** Quantify the privacy-utility trade-offs in multiagent "coordinator-worker" topologies, focusing on how data minimization principles can be enforced without degrading task performance.²⁰
- **Validate Mitigation Strategies:** Systematically benchmark the effectiveness of Randomized Masked Finetuning (RMFT) and differential privacy weightflatness optimization in reducing PII extraction rates while maintaining perplexity thresholds.²⁷
- **Establish Reproducible Evaluation:** Utilize state-of-the-art benchmarks such as PII-Scope and AgentLeak to establish a baseline for 2026-standard security auditing.²⁰

Research Scope

This research focuses on the behavior of large-scale autoregressive models, including both open-weight transformer architectures (e.g., Llama 3.3, Mistral Large) and proprietary black-box APIs (e.g., GPT-5, Claude 3.5).²⁶ The domains under investigation are limited to those with high sensitivity and established regulatory oversight: healthcare (clinical notes and scheduling), finance (transaction

records and investment strategies), legal (contracts and case law), and corporate (internal communication and proprietary code).⁴

Technically, the scope encompasses:

- **Training and Fine-tuning:** Analyzing the impact of data duplication and epoch counts on memorization.⁴
- **Inference Dynamics:** Investigating the "Neural Breadcrumbs" left in transformer hidden states and attention patterns.⁹
- **Agentic Systems:** Monitoring data flows through inter-agent messages (C2), tool inputs/outputs (C3/C4), and shared memory (C5).²⁰ The study excludes infrastructure-level attacks like hardware side-channels (e.g., GPU memory timing) except where they directly facilitate the inference of confidential system prompts.³⁶

Chapter 2: Literature Review

The Mechanism of Memorization and Extraction

The foundational research into data leakage began with the discovery that LLMs do not merely learn linguistic rules but also store massive quantities of raw text. Carlini et al. (2021) demonstrated that the probability of a model memorizing a sequence is tied to the number of times it appears in the training data, a phenomenon known as the duplication effect.¹⁰ By 2026, it has been established that sequences repeated ten times are, on average, a thousand times more likely to be memorized than those seen once.³⁴ This memorization is particularly acute for "outlier" samples—data points that are statistically distant from the rest of the corpus, such as specific names, phone numbers, or social security numbers.³⁷

The Role of Model Scale and Epochs

The relationship between model size and leakage is complex. While neural abilities generally increase with model size, larger models also have a greater capacity for memorization.³⁴ However, research on Chinchilla-optimal models (trained for a single epoch) shows a non-monotonic relationship: larger models are not always more vulnerable to membership inference, as the signal often becomes more "distributed" across parameters.³⁹ In contrast, fine-tuning on small, specialized datasets significantly elevates memorization rates, as the model "overfits" to the specific patterns of the new data.⁴

Membership Inference Attacks (MIA)

MIAs aim to determine whether a specific record was part of a model's training set.¹¹ Traditional MIAs relied on "shadow models"—multiple models trained on similar data to capture the difference in performance between member and non-member sequences.¹¹ However, these are difficult to scale for LLMs. By 2026, researchers have introduced reference-free techniques:

- **Min-K% Prob:** This method operates on the hypothesis that unseen examples contain a few outlier words with low probabilities under the model, while memorized sequences consist of high-probability tokens across the entire span.¹¹
- **Neural Breadcrumbs (memTrace):** This methodology analyzes the model's internal representations (transformer hidden states and attention patterns) rather than just final outputs. It has achieved average AUC scores of 0.85 on popular benchmarks by identifying the distinct "fingerprint" of the model's processing behavior when encountering familiar data.⁹

PII Leakage and Targeted Extraction

Personally Identifiable Information (PII) extraction represents the most critical privacy threat. PII-Scope, a comprehensive benchmark introduced in 2025, standardized the evaluation of these attacks across black-box and white-box settings.²⁹

Hyperparameter Sensitivity in PII Attacks

PII-Scope revealed that the effectiveness of extraction is highly sensitive to hyperparameters:

- **Prefix Length:** Using the "true prefix" from training data significantly increases success rates; however, if the prefix is too short, the model may default to generic or synthetic completions.³⁰
- **Demonstration Selection:** In few-shot prompting, selecting examples that are "in-domain" (related to the target) can improve PII extractability by over tenfold.⁴⁰
- **Temperature and Decoding:** Lower temperature settings (near zero) favor the model's most likely (often memorized) tokens, making extraction more stable.³⁰

A comparison of PII leakage success across various 2026-era models using a standard multi-query attack budget is presented below.

Model Name	Parameter Count	PII Extraction Rate (Single-Query)	PII Extraction Rate (Multi-Query Budget)	Success Improvement
Llama 3.3	70B	9.4%	46.2%	4.9x ²⁹
Mistral Large	123B	11.2%	51.5%	4.6x ²⁹
Claude 3.5 Sonnet	Black-Box	4.8%	22.1%	4.6x ²⁶
GPT-5-mini	Black-Box	3.5%	18.9%	5.4x ²⁹

These findings suggest that "safety-aligned" models (like GPT-5) achieve lower base leakage rates but remain susceptible to advanced adversarial strategies that use repeated and diverse querying to circumvent filters.²⁹

Benchmark Contamination and the "Leakage Trap"

The reliability of LLM evaluation is increasingly compromised by benchmark data leakage—the inclusion of test sets in pre-training data.⁴³ This phenomenon is

particularly problematic in recommendation systems, where models are expected to understand latent user interests.

The Dual-Effect of Recommendation Leakage

Research identifies a "dual-effect" of contamination:

1. **In-Domain Leakage:** When the leaked data is relevant to the recommendation task, it induces spurious performance gains. For example, on the APPS benchmark, StarCoder-7B showed a Pass@1 score 4.9 times higher on leaked samples compared to non-leaked data.⁴⁵
2. **Out-of-Domain Leakage:** Paradoxically, contamination with irrelevant benchmark data typically degrades accuracy by distorting the model's learning of item representations and user intents.⁴³

To quantify this, researchers use the performance change metric:

$$\Delta m = \frac{Perf_m(\theta_{dirty}) - Perf_m(\theta_{clean})}{Perf_m(\theta_{clean})} \times 100\%$$

Where θ_{dirty}

10

represents a model exposed to benchmark data via fine-tuning. This "leakage trap" masks the true capability of the model and rewards data exposure over architectural innovation.⁴⁶

Preference Leakage in Judge-Based Evaluation

The paradigm of "LLM-as-a-judge" has emerged to handle the evaluation of longform, open-ended generation in a cost-efficient manner.⁴⁷ However, this has introduced a subtle bias known as "Preference Leakage".⁴⁸

This issue arises when the data generator LLM (used to create synthetic training data) and the judge LLM (used to evaluate responses) are closely related. Research defines three types of relatedness:

- **Same Model:** The generator and judge are the same model (e.g., both GPT-4o).⁴⁸

- **Inheritance Relationship:** The student model is distilled or fine-tuned directly from the data generator.⁴⁸
- **Same Model Family:** Both models belong to the same developer family (e.g., Llama 2 and Llama 3).⁴⁸

Experimental results confirm that judge LLMs exhibit a systematic bias toward their related student models because they recognize and favor their own inherent stylistic and logical "preferences" in the output.⁴⁸ This bias is subtler than egocentric bias (favoring one's own output) and is significantly harder to detect because most LLM providers do not disclose their training or distillation histories.⁴⁸

Agentic AI and Full-Stack Leakage Pathways

The transition toward autonomous agents—systems that can set goals, use tools, and maintain persistent memory—has created a "full-stack" privacy risk.²⁰ The AgentLeak benchmark, introduced in 2026, instruments the data flows of multiagent systems to identify where information escapes.²⁰

The Seven Leakage Channels (Taxonomy)

AgentLeak identifies seven distinct pathways through which sensitive data propagates²⁰:

Channel ID	Channel Name	Category	Description
C1	Final Output	External	Data presented directly to the user; the primary target of safety filters. ²⁰

C2	Inter-agent Messages	Internal	Communication between agents during task delegation; the most frequent leakage point. ²⁰
C3	Tool Inputs	External	Arguments passed to external APIs (e.g., medical scheduling endpoints). ³⁵
C4	Tool Outputs	External	Sensitive data returned from external systems and stored in context. ²⁰
C5	Shared Memory	Internal	Persistent state across execution boundaries; enables cross-session leakage. ²⁰
C6	Telemetry/Logs	External	Traces and logs stored for monitoring; often quietly accumulate PII. ²⁰

C7	Artifacts	External	Generated files, reports, or records stored on disk. ²⁰
-----------	-----------	----------	--

Coordinator-Worker Topology Vulnerabilities

In multi-agent configurations, researchers found a troubling pattern: while perchannel output leakage (C1) often decreases (as agents are more specialized), the total system exposure increases due to internal channels.²⁰ For instance, a scheduling agent might return a clean, privacy-compliant confirmation to a user, while its internal delegation message to a verification agent (C2) carries the patient’s complete, unmasked medical record.²⁰ Internal channels show 2.1x higher leak rates than external ones, with C2 (messages) reaching leakage rates of 68.8%—meaning output-only auditing misses approximately 41.7% of all privacy violations in agentic workflows.²¹

State-of-the-Art Mitigation and Defense

The research community has proposed several layers of technical and governancebased defenses to counter these evolving threats.

Randomized Masked Finetuning (RMFT)

RMFT is a training-time intervention designed to mitigate PII memorization.²⁸ It identifies all occurrences of sensitive fields (e.g., email addresses) in a dataset and preserves only the first occurrence in its original form. Subsequent occurrences are replaced with randomized variants that are syntactically valid but semantically different.²⁸ This prevents the model from assigning high probability to a specific private string while still allowing it to learn the general structure of the data.²⁸

Differential Privacy and Weight Flatness

Differential Privacy (DP) adds noise to gradient updates during training to ensure that the influence of any single data record is mathematically bounded.³⁶ Recent research at ICLR 2024 revealed that the "flatness" of the DP-trained model's loss landscape is essential for the utility-privacy trade-off.²⁷ By enforcing appropriate weight flatness through perturbation-aware min-max optimization, researchers have

achieved generalization performance comparable to non-private fine-tuning under a privacy budget of $\epsilon = 3$.²⁷

Privacy-Preserving Parameter-Efficient Fine-Tuning (RAPT)

In scenarios where users fine-tune models via third-party APIs (e.g., OpenAI or Google Cloud), the RAPT framework offers a local privacy approach.⁵³ Users privatize their data locally using a text-to-text local DP mechanism. To counter the performance drop typically seen with privatized data, RAPT introduces a "privatized token reconstruction" task that is trained jointly with the downstream task, helping the model learn better representations of the noise-obscured data.⁵³

Post-Training Machine Unlearning

When specific data must be removed to comply with legal mandates, models undergo unlearning.²³ Techniques like Gradient Ascent (GA) suppress the knowledge of a "forget set" by moving parameters away from the high-probability region of those tokens.²³ To prevent catastrophic forgetting of general knowledge, GA is often combined with KL Divergence Minimization (KLR), which constrains the unlearned model's output distribution to remain close to the original "safe" model.²³

Chapter 3: Research Methodology

Proposed Methodology: Sentinel AI Framework

This chapter outlines the proposed methodology for developing **Sentinel AI**, a comprehensive, multi-tiered monitoring and response system designed to detect and block data leakage in real-time within production environments. Sentinel AI moves beyond static filters by incorporating behavioral analysis and internal channel instrumentation.²⁵

1. Tiered Detection Pipeline

The system will implement a three-tier unified detection mechanism across all seven leakage channels identified in the AgentLeak taxonomy.²⁰

- **Tier 1: Canary Matching:** The system inserts unique, high-entropy markers ("canaries") into sensitive data sources. A high-confidence regex-based matching engine monitors all seven channels for the verbatim appearance of these markers, providing a baseline validation that the capture infrastructure is functional.²⁰

- **Tier 2: Pattern Extraction and Validation:** This tier uses format-valid fake identifiers (e.g., strings that satisfy the Luhn checksum for credit cards or the three-two-four digit format for SSNs). An Isolation Forest algorithm—a lightweight, non-linear ML model—identifies anomalous query structures or packet sizes that suggest slow, consistent data exfiltration.¹⁸
- **Tier 3: LLM-as-a-Judge Reasoning:** For contextually sensitive data that cannot be captured by regex (e.g., "John is switching jobs"), the system employs a reasoning-capable judge model (e.g., GPT-5-mini). This judge evaluates the "semantic relevancy" of data flows against the allowed disclosure set defined for a specific task.²⁰

2. Experimental Setup and Benchmarking

To validate the Sentinel AI framework, the project will use a mixed-method empirical approach.

- **Datasets:** Training and validation will utilize the Enron Email Dataset (raw version) for PII extraction testing and WikiText for baseline language modeling perplexity.²⁸
- **Target Models:** The system will be tested across a factorial evaluation crossing five production models: GPT-4o, Claude 3.5 Sonnet, Mistral Large, Llama 3.3 70B, and GPT-4o-mini.²⁶
- **Adversarial Probing:** The Garak (NVIDIA) and PyRIT (Microsoft) redteaming toolkits will be used to generate over 100 attack probes, covering encoding tricks, jailbreaks, and RAG poisoning.¹⁷

3. Metrics for Quantification

The primary metrics for evaluating the system's effectiveness include:

- **Exact Leakage Rate (ELR):** The percentage of scenarios where at least one sensitive field leaked.²⁰
- **Weighted Leakage Score (WLS):** A severity-weighted sum of leaks based on the sensitivity of the field (e.g., SSN carries more weight than job title).²⁰
- **MaxTER (Max Training Extraction Reduction):** A metric defining the optimal checkpoint where TER (Training Extraction Rate) reduction is

maximized while keeping MDP (Model Perplexity increase) below a 20% threshold.²⁸

Expected Outcomes

The Sentinel AI project is expected to deliver the following artifacts:

1. **Runtime Defense Engine:** A functional software package compatible with LangChain and MCP (Model Context Protocol) that provides real-time visibility and policy control for inter-agent workflows.²²
2. **Privacy-Optimized Model Adapters:** A set of LoRA adapters fine-tuned using RMFT and weight-flatness DP, demonstrating a "Pareto optimal" tradeoff between task completion and privacy.²⁰
3. **Audit Evidence Framework:** An automated logging system that generates immutable traces of context, actions, and outcomes to support GDPR and HIPAA compliance attestations.⁴
4. **PII-Compass Dashboard:** A visual interface for security analysts to monitor "semantic drift" and exfiltration risk scores across distributed LLM nodes.²⁵

Project Plan and Gantt Chart

The development of the Sentinel AI framework is scheduled over a 12-month period, structured around five distinct project phases.⁵⁹

Phase 1: Requirements and Domain Analysis (Months 1-2)

Tasks include identifying key stakeholders, defining the SMART goals (Specific, Measurable, Achievable, Relevant, Time-bound), and understanding the aviation/healthcare/finance-specific regulatory constraints.⁶²

- **Milestone:** Finalized project scope and success metrics.

Phase 2: Design and Architecture (Months 3-4)

Tasks involve mapping out the inter-agent coordinator-worker topologies and designing the data-flow call graphs.²⁰

- **Milestone:** Architectural diagrams and protocol specifications completed.

Phase 3: Development and Fine-Tuning (Months 5-7)

Tasks include implementing the RMFT training pipeline and the tiered detection filters.²⁵

- **Milestone:** Sentinel AI prototype ready for integration.

Phase 4: Adversarial Testing and Red-Teaming (Months 8-10)

Tasks involve using Garak and PyRIT to simulate multi-turn attacks and validating safeguards under a changing context.⁷

- **Milestone:** Security audit report and model vulnerability assessment.

Phase 5: Deployment and Finalization (Months 11-12)

Tasks include final system optimization, user training, and the release of opensource components.⁶⁰

- **Milestone:** Final project delivery and stakeholder presentation.

12-Month Implementation Timeline (Gantt Chart)

[illegible]

Task 2.0 focus on inter-agent messages (C2) and shared memory (C5) as per AgentLeak insights.²⁰ Task 4.0 uses MaxTER for checkpoint optimization.²⁸

Strategic Implications and Future Directions

The research findings indicate that data leakage in LLMs is shifting from a technical bug to a dominant operational risk. By 2026, leading organizations are treating LLM interactions as "privileged interfaces" requiring explicit controls and auditing.⁷ The key emerging benchmark is runtime behavior over static design—testing how models behave under adversarial input across real workflows.⁷

Critical research gaps remain in "private inference for long contexts" and the integration of leakage detection with sparse Mixture-of-Experts (MoE) architectures, where routing logic may itself reveal information about the input data.³⁶ Additionally, as models move toward multi-modal and agentic autonomy, the need for "zero-trust" AI governance becomes paramount, where no agent assumes the safety of its peers or its tools.¹⁷ The Sentinel AI framework establishes a vital foundation for this shift, providing the first steps toward a verifiable, privacy-preserving ecosystem for autonomous artificial intelligence.

Reference

1. Anonymous. (2026). *Randomized Masked Finetuning for PII Mitigation in LLMs*. GitHub Repository.
2. Baykaloğlu, Y. (2026). *Multi-Agent Systems: Orchestrating AI Agents with A2A Protocol*. Online Inference/Medium.
3. Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K.,... & Raffel, C. (2021). *Extracting Training Data from Large Language Models*. USENIX Security Symposium.
4. Chen, J., et al. (2024). *LLMs in Software Security: A Survey of Vulnerability Detection Techniques and Insights*. arXiv preprint.
5. Duan, J., et al. (2024). *Do Membership Inference Attacks Work on Large Language Models?* arXiv preprint arXiv:2505.18773.
6. El Du, E., Dwork, C., Tankala, P., & Zhang, L. (2026). *Differentially Private Verification of Distribution Properties*. arXiv preprint.

7. El Yagoubi, F., Badu-Marfo, G., & Al Mallah, R. (2026). *AgentLeak: A FullStack Benchmark for Privacy Leakage in Multi-Agent LLM Systems*. arXiv preprint arXiv:26 02.11510.
8. Ertan, B., et al. (2026). *On the Effectiveness of Membership Inference in Targeted Data Extraction*. Hugging Face Papers.
9. Huang, Y., et al. (2026). *Building a Foundational Guardrail for General Agentic Systems via Synthetic Data*. ICLR 2026.
10. Juneja, A., et al. (2025). *MAGPIE: A Contextual Privacy Benchmark for Multi-Agent Systems*. arXiv preprint.
11. Li, D., Sun, R., Huang, Y., Zhong, M., Jiang, B., Han, J., Zhang, X., Wei, W., & Liu, H. (2026). *Preference Leakage: A Contamination Problem in LLM-as-a-judge*. The Fourteenth International Conference on Learning Representations (ICLR 2026).
12. Liu, Z., et al. (2025). *Secure Multi-LLM Agentic AI and Agentification for Edge General Intelligence by Zero-Trust*. arXiv preprint.
13. Maini, P., et al. (2024). *Gradient Descent on the Retain Set (GDR) for Machine Unlearning*. arXiv preprint.
14. Meeus, M., et al. (2024). *Review of existing MIA benchmarks and distribution shifts*. arXiv preprint.
15. Nakka, K. K., & Frikha, A. (2024). *PII-Compass: Guiding LLM training data extraction*. Proceedings of the Association for Computational Linguistics (ACL).
16. Nakka, K. K., Frikha, A., Mendes, R., Jiang, X., & Zhou, X. (2025). *PIIScope: A Comprehensive Study on Training Data Privacy Leakage in Pretrained LLMs*. IJCNLP-AAACL 2025.
17. Nakka, K. K., et al. (2026). *Unintended Memorization of Sensitive Information in Fine-Tuned Language Models*. arXiv preprint arXiv:2601.17480.
18. Nie, X., et al. (2024). *PrivAgent: An Agentic Red-Teaming framework for detecting systemic privacy vulnerabilities*. arXiv preprint.
19. Nolte, H., Finck, M., & Meding, K. (2025). *Machine Learners Should Acknowledge the Legal Implications of LLMs as Personal Data*. arXiv preprint.
20. OWASP Foundation. (2025). *OWASP Top 10 for LLM Applications 2025*.
21. Pasch, A., & Cha, H. (2025). *Anonymization effectiveness in LLM privacy leakage mitigation*. arXiv preprint.

- 22.Sahili, A., Chehab, A., & Tajeddine, R. (2025). *On the Effectiveness of Membership Inference in Targeted Data Extraction*. arXiv preprint.
- 23.Shi, W., et al. (2024). *Min-K% Prob: Localized Token-level Memorization in LLMs*. EMNLP 2024.
- 24.Stauber, D., Morehouse, K. N., Hartmann, D., & Berendt, B. (2026). *HumanCentred LLM Privacy Audits: Findings and Frictions*. arXiv preprint arXiv:2603.12094.
- 25.Sun, Y., et al. (2024). *Privacy-preserving Fine-tuning of Large Language Models through Flatness*. ICLR 2024 Workshop.
- 26.Watanabe, T., et al. (2026). *Slop: Maintenance accumulation in traditional software engineering*. arXiv preprint.
- 27.Xu, M., et al. (2024). *On Protecting the Data Privacy of Large Language Models (LLMs): A Survey*. arXiv preprint.
- 28.Yoon, J., et al. (2026). *Towards Privacy-Preserving Large Language Model: Text-free Inference Through Alignment and Adaptation*. arXiv preprint arXiv:2604.00761.
- 29.Zhang, M., Peng, Q., Wang, Y., Liu, C., & Liu, H. (2026). *Benchmark Leakage Trap: Can We Trust LLM-based Recommendation?* arXiv preprint arXiv:2602.13626.
- 30.Zhou, X., Weyssow, M., et al. (2026). *LessLeak-Bench: A First Investigation of Data Leakage in LLMs Across 83 Software Engineering Benchmarks*. Computers, 15(4), 226.