What is training data poisoning?

Training data poisoning is when someone slips adversarial content into the data an AI learns from, so the trained AI later misbehaves on specific inputs. Three failure modes: hidden triggers that flip the AI into bad behavior on a specific phrase, targeted wrong-answers on attacker-chosen inputs, and AI outputs that reveal sensitive content that was in the training data. This is different from RAG poisoning, which attacks the AI's reading material at query time rather than its training.

HOW IT WORKS

01 How is training data poisoning different from RAG poisoning?

Training data poisoning attacks the model's weights. RAG poisoning attacks the retrieval corpus the model reads at inference time. The defenses look very different.

- Training poisoning is fixed by changes to the training pipeline (data hygiene, provenance, deduplication, differential privacy) before the model ships. Once the weights are out, the only fix is retraining.
- RAG poisoning is fixed at inference time by changes to ingestion, retrieval, and output verification. The model itself does not need to change.

Most LLM-backed applications today use a foundation model plus RAG; for those, RAG poisoning is the much more likely attack. Training poisoning matters most when you fine-tune your own model on data with mixed provenance, when you depend on a foundation model trained on web-scraped data of unknown integrity, or when the model's training set contained sensitive records you do not want extractable. See our RAG poisoning explainer for the inference-time variant.

02 What are the three main failure modes?

Targeted misclassification. The model produces the wrong answer for specific inputs but behaves normally everywhere else. Benchmarks miss it because benchmarks do not contain the attacker's specific triggers. Documented in classifier-poisoning research since the early 2010s

HOW TO DEFEND

- Data provenance. Track where every training sample came from. Reject samples from sources you cannot trace.
- Deduplication. Memorization risk scales with how many copies of a sample appear in training. Aggressive deduplication is the cheapest single defense against verbatim leakage.
- Anomaly screening. Statistical tests for unusual sample distributions, near-duplicate topic areas from a single source, or trigger-shaped phrasing. Catches some poisoning, misses sophisticated attacks.
- Differential privacy in training. Adds calibrated noise that bounds how much any single training sample can influence the model. Comes with a quality tradeoff; appropriate when training-data privacy is a hard requirement.
- Reduced reliance on uncurated web data. Training on smaller, more carefully curated datasets is a structural defense. Most production-grade fine-tuning already does this.
- Post-training probing. Test the model against suspected triggers and against extraction prompts before shipping. Cannot catch everything; raises the cost of undetected poisoning.

SOURCES

and adapted to generative models in the past two years.

Backdoor triggers. A specific input string (a phrase, a token sequence, a piece of code) reliably switches the model into a different behavior. The classic example is BadNets (Gu et al., 2017), which embedded a backdoor trigger in image classifiers. The LLM-era equivalents embed text triggers that flip generation behavior or unlock disallowed responses.

Memorized leakage. The model emits verbatim chunks of its training data when prompted in the right way. Carlini et al. (2021) showed GPT-2 would recite real names, addresses, and code snippets from its training set. The risk is most acute when training data contains regulated personal information, secrets, or proprietary content.

03 Are these attacks practical today?

Yes, with caveats. Practical demonstrations exist in the academic literature for each failure mode. Real-world incidents are harder to attribute publicly (poisoning is by nature subtle and most affected organizations would not announce it).

- Web-scraped data is the easiest poisoning channel. Anyone who can get content indexed by a major web crawler can attempt to inject training samples. Carlini et al. (2024) showed a single attacker could inject tens of thousands of poisoned samples into common training corpora for under $100.
- Open-source dataset publication is a supply-chain risk. A popular Hugging Face dataset compromised by a malicious contributor can poison every model that fine-tunes on it.
- Crowd-sourced labeling is a vector. Mechanical Turk and similar pipelines can be infiltrated by attackers willing to label many samples cheaply.
- RLHF preference data is the newest channel. Adversarial labelers in a preference-collection pipeline can teach the model preferences that subtly bias its outputs.

04 How does SecureLayer7 test for training data poisoning risk?

[1] OWASP LLM04:2025, Data and Model Poisoning
[2] Gu et al., BadNets (backdoor attacks on neural networks)
[3] Carlini et al., Extracting Training Data from Large Language Models
[4] Carlini et al., Poisoning Web-Scale Training Datasets is Practical

Three things on every AI engagement that touches a custom-trained or fine-tuned model.

Provenance audit. We map every training data source the team uses, score each on attacker reachability, and flag the highest-risk channels for additional hardening.

Backdoor probing. We test the deployed model against suspected trigger patterns drawn from prior incidents in the literature and from the team's own data sources.

Memorization extraction. We run extraction prompts against the model targeting whatever sensitive content might have been in the training set (customer records, internal docs, secrets). We document any extractable content and the prompt structure that surfaced it.

Deliverable maps findings to OWASP LLM04 and includes recommended pipeline changes prioritized by attacker effort.

Audit your training and fine-tuning pipeline.
securelayer7.net/learn/ai-security/training-data-poisoning

Open online