

# Agent Identity Is Not Model Identity

*Why authenticating the software is not the same as proving which model is actually computing*

Anthony Ray Coslett\*

Fall Risk Research

March 2026

A modern AI deployment stack typically answers two identity questions well.

The first is artifact identity. Model registries, version-controlled repositories, and bills of materials record which files were deployed at a given point in time. The artifact question — *what was shipped?* — has mature tooling and well-understood governance.

The second is agent identity. OAuth tokens, API keys, SPIFFE SVIDs, mTLS certificates, and workload attestation frameworks authenticate which software is authorized to act. The agent question — *what is making this request?* — is the focus of significant current investment. Okta’s AI agent identity suite [5], Microsoft Entra’s non-human identity capabilities [8], and the NIST NCCoE concept paper on software and AI agent identity [4] all address this layer.

These are real and necessary controls. But they answer a question about the software harness — the orchestration code, the API gateway, the service mesh identity. They do not answer a question about the neural network inside it.

A third question remains:

*Which model is actually producing outputs at inference time?*

This question is distinct from the first two. An open-weight model can be fine-tuned, distilled, quantized, composed into routing pipelines, or silently swapped behind a stable endpoint. When any of these occur, the artifact record may become stale, but the agent credentials remain valid. The workload identity doesn’t change. The API key still authenticates. The service mesh still routes. The OAuth token still authorizes. Every identity control in the stack continues to function correctly — because none of them were verifying the model. They were verifying the software that hosts it.

The distinction is not hypothetical. In March 2026, a commercially deployed AI coding tool launched a flagship model described as the product of continued pretraining and reinforcement learning [6]. Days after launch, a developer examining the tool’s API responses discovered an internal model identifier referencing a different organization’s open-weight model. The deployer subsequently acknowledged the foundation. The agent identity layer — credentials, API routing, workload authentication — had been functioning correctly throughout. The model foundation was established through an accidental metadata string in a debug response, not through any runtime identity check.

In the adjacent domain of software supply chains, a parallel incident demonstrated the same structural gap [7]. A poisoned package affecting a widely deployed AI model-routing library was published to PyPI using legitimate credentials. The package passed every standard integrity check: the file matched what the registry advertised, the hash was correctly declared, and `pip install --require-hashes` would have succeeded. As the security team that analyzed it observed: hash verification confirms that a file matches what was advertised, but it does not indicate whether the advertised content is safe. The artifact was intact. The content was not what it appeared to be.

Both incidents share the same category error: integrity evidence was treated as identity evidence. Knowing that an artifact matches its record is not the same as knowing what it will execute. Knowing that a workload is authenticated is not the same as knowing which model the workload is running.

---

\*Correspondence: [anthony@fallrisk.ai](mailto:anthony@fallrisk.ai).

This suggests a simple taxonomy [1]. Four questions define the identity surface of an AI system in deployment:

1. *What artifact was shipped?* — answered by registries, version control, bills of materials.
2. *What agent or workload is authenticated?* — answered by OAuth, SPIFFE, Okta, Entra, mTLS.
3. *What model is actually computing?* — **not answered by any widely deployed system, as publicly described.**
4. *Whose training lineage is present?* — a forensic extension addressing provenance and distillation history.

Current infrastructure covers questions one and two thoroughly. Questions three and four occupy a different evidence class — one that requires observing the model during computation rather than inspecting credentials or artifacts at rest.

Recent research demonstrates that this third question is technically answerable [2]. A structural measurement based on activation geometry during a standard forward pass can identify a specific neural network at inference time, without inspecting weights, modifying the model, or requiring cooperation from the deployer. This capability has been validated on open-weight transformer models spanning 410 million to 72.7 billion parameters, including distilled derivatives that share identical architecture and parameter count with their stated base models. The identity result can be issued as a signed software attestation and consumed by the same policy engines, service identity frameworks, and gateway enforcement systems that enterprises already operate [3]. Prior work also validates stronger trust modes in which the identity measurement is hardware-attested and bound to a specific output claim, providing a root of trust beneath the software attestation layer [3]. The measurement does not replace agent or workload identity. It occupies a different layer — the layer that proves which model is actually running.

The question of whether this layer matters is, in practice, a question about what an organization needs to trust. If the governance requirement is “authenticate which software made this request,” current agent identity frameworks are sufficient. If the requirement extends to “confirm which model produced this output” — for regulatory compliance, supply chain assurance, contractual verification, or internal policy enforcement — then agent identity alone leaves the question unanswered. The software identity layer can work perfectly while the model identity layer is entirely absent.

The agent is the software that acts. The model is the computation that decides. Authenticating one has never been equivalent to verifying the other.

## References

- [1] A. R. Coslett. What counts as proof? — Admissible evidence for neural network identity claims. *Zenodo*, 2026. DOI: 10.5281/zenodo.19058540.
- [2] A. R. Coslett. Post-hoc disclosure is not runtime proof: Model identity at frontier scale. *Zenodo*, 2026. DOI: 10.5281/zenodo.19216634.
- [3] A. R. Coslett. Composable model identity — Formal hardening of structural attestations in the enterprise identity stack. *Zenodo*, 2026. DOI: 10.5281/zenodo.19099911.
- [4] NIST NCCoE. New concept paper on identity and authority of software agents. 2026. <https://www.nccoe.nist.gov/news-insights/new-concept-paper-identity-and-authority-software-agents>.
- [5] Okta. Okta for AI agents. Announced March 2026, GA April 30, 2026. <https://www.okta.com/newsroom/press-releases/showcase-2026/>.
- [6] A. Ha. Cursor admits its new coding model was built on top of Moonshot AI’s Kimi. *TechCrunch*, March 22, 2026. <https://techcrunch.com/2026/03/22/cursor-admits-its-new-coding-model-was-built-on-top-of-moonshot-ais-kimi/>.
- [7] Snyk Security Research. How a poisoned security scanner became the key to backdooring LiteLLM. March 2026. <https://snyk.io/articles/poisoned-security-scanner-backdooring-litellm/>.
- [8] Microsoft. What is Microsoft Entra Agent ID? 2026. <https://learn.microsoft.com/en-us/entra/agent-id/identity-professional/microsoft-entra-agent-identities-for-ai-agents>.