Proceedings of GICITE, Berlin, Germany (6-7th November, 2026)

WINS *Proceedings*

ISSN: 3153-8304

DOI
10.XXXX/ICXXX.2027.000

Proceeding Paper

AI Supply Chain Security and Model Provenance: Verifiable Trust from Training Data to Agentic Runtime

Srihari Pakalapati

Trader Interactive (CAR Group)

United States of America

sriharipakalapati@gmail.com

***Abstract*—**AI security has concentrated on inference-time threats such as prompt injection and jailbreaks. A deeper attack surface lies beneath the model: the supply chain of artifacts that determines its behavior-training data, weights, pipelines, dependencies, and, in agentic systems, the tool schemas consumed at runtime. This paper examines AI supply chain security through the lens of model provenance. It assesses which conventional software supply chain controls carry over and which fail, and adapts SLSA provenance levels, Sigstore-based model signing, in-toto attestations, and AI bills of materials (AI-BOM) to the artifacts that shape model behavior, including agentic protocols such as the Model Context Protocol (MCP) that move composition to inference time. A reference architecture and a five-gate admission framework are proposed.

***Keywords*—**AI supply chain security, model provenance, model signing, SLSA, AI bill of materials, agentic AI, Model Context Protocol

Published: 6 November 2026

I. Introduction

Today's AI systems are constructed by assembling disparate parts-public foundation models, obscure adapters, massive datasets, and dynamic agentic tools-rather than writing code from scratch. This reliance on external supply chains introduces a critical security gap: almost none of these components are cryptographically verified. Even engineering teams with strict traditional security practices routinely deploy massive, unattested AI checkpoints. Driven by active threats like data [1] and tool poisoning [6], as well as new regulatory mandates from the EU AI Act [7] and NIST AI RMF [5], this paper argues for a paradigm shift. We propose that any element altering model behavior must be treated as a securely signed, attested, and inventoried artifact, replacing static documentation with an unbroken chain of cryptographic evidence.

The argument proceeds in four steps. The paper first describes the layers of an AI system as a supply chain and explains why the agentic layer behaves differently from the rest. It then reviews the main threat classes and the degree to which conventional controls cover them. Next, it describes four mechanisms that together give a model verifiable provenance, and finally it describes runtime controls and a five-gate process for admitting third-party components. The aim throughout is practical. Each mechanism is chosen because it adapts tooling that software teams already operate, so adoption can start from existing signing, build and inventory infrastructure and does not require new cryptographic methods.

II. The AI Supply Chain and the Agentic Shift

An AI system can be viewed as five interconnected supply-chain layers, each introducing a distinct trust boundary: data, including training, preference, and retrieval corpora; models, such as weights, checkpoints, and adapters; pipelines spanning training code, CI/CD, and evaluation; dependencies including frameworks, accelerator software, and inference infrastructure; and finally, tools and agents, including MCP servers, tool definitions, and externally returned context. This fifth layer challenges two assumptions embedded in traditional supply-chain security. First, composition is no longer confined to build time: agents may discover and incorporate new tools dynamically at runtime, bypassing conventional deployment and pipeline controls. Second, execution is not fully deterministic. Because language models can be influenced through natural-language inputs, even a legitimate tool may become an exfiltration path when manipulated descriptions or context steer model behavior. In such systems, trust must therefore be evaluated across artifact integrity, runtime execution, and semantic behavior, whereas classical supply-chain controls primarily address the artifact layer.

Fig. 1 depicts the five layers and the shift introduced by the fifth.

The layers are also connected. A weakness in one layer can be inherited by the next, and the effect often becomes visible only far from its origin. A poisoned corpus shapes the weights trained on it; those weights are fine-tuned with an adapter obtained from a different source; the result is served through a framework with its own dependency tree; and the served model is finally given tools that act on production systems. At each hand-off the receiving team usually sees a finished artifact and not the process that produced it, so it has no basis for judging what was inherited. This is why a single control at one layer, however strong, cannot establish trust in the whole system. It is also why records of what was used at each stage, kept in a form that can be checked, matter as much as the controls themselves.

Ownership adds a second difficulty. The five layers are often managed by different groups: data engineering owns the corpora, a machine learning platform team owns training and serving, a security group owns the dependencies and the build system, and application teams own the agents and tools. Each group may meet its own obligations and yet leave gaps at the boundaries, for example when no group is responsible for confirming that the weights deployed are the ones that were evalu-

© 2026 The Authors. Published in the Proceedings of GICITE 2026.

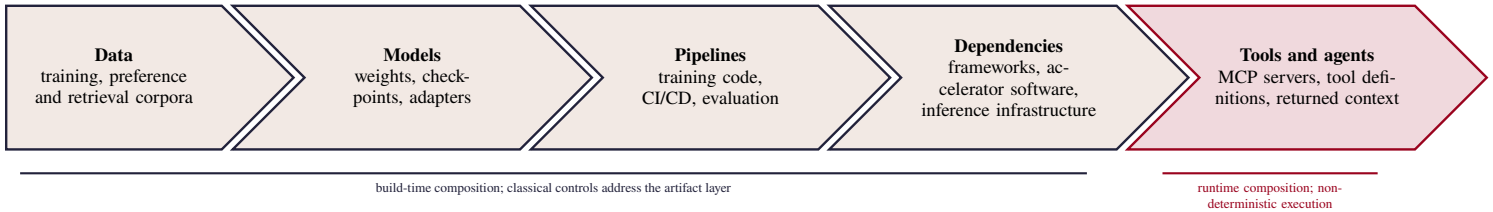

Fig. 1. The five supply-chain layers of an AI system, each introducing a distinct trust boundary. The tools-and-agents layer requires trust to be evaluated across artifact integrity, runtime execution and semantic behavior.

TABLE I
ASSUMPTIONS OF TRADITIONAL SUPPLY-CHAIN SECURITY CHALLENGED BY THE TOOLS-AND-AGENTS LAYER

Assumption	In agentic systems	Implication
Composition is confined to build time	Agents discover and incorporate tools at runtime, bypassing deployment and pipeline controls	The control plane must extend into the runtime
Execution is deterministic	Natural-language inputs can steer behavior; a legitimate tool can become an exfiltration path	Trust must cover semantic behavior, not only artifacts

ated. A provenance chain gives these groups a shared object to reason about, since each hand-off can be recorded as a signed statement that the next group verifies.

III. Threat Landscape and Conventional Controls

Table II consolidates the principal threat classes by layer. The most likely threats malicious hub artifacts, dependency confusion, credential theft from over-privileged connectors, and silent upstream model swaps-are classical attacks in an AI setting with mature mitigations. Threats novel to AI, such as trigger-conditioned backdoors and tool poisoning, are less likely today but are precisely those existing tooling cannot detect. In recurring incidents, source review passed and the perimeter held, yet the artifact was hostile; verification must therefore move onto the artifact itself.

Table I sets out the two assumptions discussed above.

Two further points help in reading the threat table. The first is that the status of a conventional control depends on what it is able to examine. Software composition analysis can list a vulnerable library, but it cannot say whether a set of weights has been altered to respond to a hidden trigger, because the relevant behavior is encoded in numbers and not in code that can be reviewed. The second is that likelihood and detectability pull in opposite directions. The attacks that are most common today, such as a malicious package or a look-alike repository, are the ones for which scanners and policies already exist. The attacks that are hardest to detect, such as a targeted backdoor or a manipulated tool description, are the ones an organization is least likely to have tested for. A sound program therefore does not rank threats by frequency alone. It asks which failures would go unnoticed and arranges for verification at the points where that is possible.

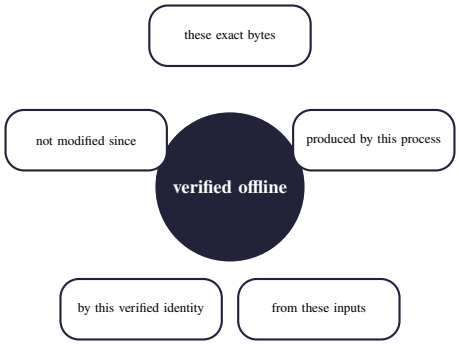

Fig. 2. What a verifier must be able to check, without trusting the distribution channel, for a model to have provenance.

IV. Model Provenance: A Verifiable Chain of Trust

Model cards and registry metadata are documentation, trivially forgeable. A model has provenance when a verifier can check, offline and without trusting the distribution channel, that these exact bytes were produced by this process, from these inputs, by this verified identity, and have not been modified since. Four mechanisms deliver these properties.

Fig. 2 summarizes these properties.

Table III compares the four mechanisms described below.

Model signing. OpenSSF Model Signing, built on Sigstore [4], signs a manifest of content hashes over a model's files with keyless, transparency-logged signatures. Verification becomes load-time policy: weights whose signature or identity does not match do not load, closing the hub-substitution class irrespective of delivery path. Two hygiene controls are immediately deployable: prohibiting unsafe serialization in favor of safetensors, since deserializing untrusted pickle checkpoints is arbitrary code execution; and pinning artifacts by digest rather than by tag. Research on malicious code poisoning in pre-trained model hubs makes the same point from the attacker's side: it argues that model files should be treated as code, and it measures such attacks on public hubs [8]. For a deploying team, checks on the hub side cannot replace verification on the consuming side. A practical policy is to accept only models that load through formats that cannot execute code, to verify the signature and publisher identity before loading, and to record the digest of every model that enters the environment.

Lifecycle attestations. The in-toto framework [3] provides a signed envelope for each lifecycle step-ingestion, training, evaluation, deployment. Chained, they form an evidence graph a policy engine can traverse, enabling rules such as "deploy only weights with a signed safety-evaluation attestation over these exact digests."

TABLE II
PRINCIPAL THREAT CLASSES BY SUPPLY CHAIN LAYER AND STATUS OF CONVENTIONAL CONTROLS

Layer	Representative threats	Conventional control status
Data	Web-scale and split-view poisoning; retrieval-corpus injection	Not covered: no SBOM equivalent; evaluation cannot reveal targeted triggers
Model	Hub typosquatting; unsafe serialization payloads; trojaned weights; silent version swaps	Partial: signing and digest pinning transfer; static analysis cannot read weights
Pipeline	CI/CD compromise of training or server builds; poisoned base images	Covered: SSDF and SLSA practices transfer directly
Dependencies	Dependency confusion; registry poisoning; stale CVEs	Covered: SCA, allow-lists, and pinning transfer directly
Tools / agents	Tool poisoning; hidden manifest methods; post-approval behavior change; shadow servers	Not covered: composition occurs at runtime; payloads are semantic, not code

TABLE III
THE FOUR PROVENANCE MECHANISMS AND WHAT EACH LETS A VERIFIER CHECK

Mechanism	Basis	What it establishes	Immediate practice
Model signing	OpenSSF Model Signing on Sigstore [4]	These exact bytes, signed by a verified identity; mismatched weights do not load	Safetensors instead of pickle; pin artifacts by digest
Lifecycle attestations	in-toto [3]	Signed evidence for ingestion, training, evaluation and deployment, chained into a graph	Deploy only weights with a signed safety-evaluation attestation
Build integrity levels	SLSA [2]	What process produced the weights; tampering detected	Target Level 3 process provenance
AI bill of materials	SPDX 3.0 AI profiles or CycloneDX ML-BOM	Every asset that influences behavior; runtime entries for agents	Gateway emits tool entries append-only

Attestations are most useful when they are specific. A statement that an evaluation was run is weak; a statement that names the exact model digest, the evaluation suite and its version, the dataset digest and the result is one a policy engine can check. Teams should also decide who may sign each kind of statement. The group that trains a model should not be the only one able to attest that it passed safety evaluation, and the identity that signs an evaluation attestation should be distinct from the one that signs the weights. Separating these roles means that compromise of one signing identity cannot by itself produce a model that appears fully vetted.

Build integrity levels. SLSA [2] transfers to model production once a training run is treated as a build: Level 1, every run emits provenance linking the artifact to source, commit, and dataset digests; Level 2, a hosted platform signs it; Level 3, runs are hermetic and ephemeral with provenance unforgeable even by administrators; Level 4, the transitive closure is attested. Bit-identical reproducibility of large runs is not realistic; the achievable target is Level 3 process provenance, which proves what produced the weights and detects tampering.

The case for process provenance rests on a systematic record of how supply chains are attacked. A taxonomy of attacks on open-source software supply chains catalogues them across the stages from source code to distribution and use [9]. Training pipelines have the same stages, with data added, so the taxonomy offers a checklist for deciding which hand-offs in a training run need signed evidence. Hardening the build and signing steps does not by itself protect against poisoned data, but it ensures that the pipeline which consumed the data is the one the organization intended.

Table IV lists the four levels.

TABLE IV
SLSA LEVELS APPLIED TO MODEL PRODUCTION

Level	Requirement for a training run
1	Every run emits provenance linking the artifact to source, commit and dataset digests
2	A hosted platform signs the provenance
3	Hermetic, ephemeral runs; provenance unforgeable even by administrators (achievable target)
4	The transitive closure is attested

AI bill of materials. An AI-BOM extends the SBOM to every asset that influences behavior-models, datasets, pipelines, tool integrations, dependencies, system prompts, and identity assignments-using SPDX 3.0 AI profiles or CycloneDX ML-BOM. For agentic systems it must be dynamic: runtime entries-tools available and invoked-are emitted by the agent gateway and stored append-only.

An AI-BOM is only as good as the metadata it is filled from. Empirical work on the reuse of pre-trained models from a public registry examines how models are selected and what information about them is available to the reuser [10]. Where that information is incomplete, the entry should say so and not infer values. Each entry should carry the digest, source, version, license and the evaluations a component has passed, with unknown fields recorded as unknown. The inventory should be regenerated at every build and compared across deployments, so that drift between what was approved and what is running becomes visible as a difference in two documents and not as a surprise during an incident.

V. SECURING THE AGENTIC RUNTIME AND ADMITTING COMPONENTS

Because composition occurs at inference time, the control plane must extend into the runtime through six control families: transport and identity (mutual TLS, sender-constrained tokens); context integrity (strict schema validation); tool integrity (signed tool definitions so post-approval mutation is detectable, with deny-by-default scopes); gateways mediating every agent-to-

TABLE V
THE SIX RUNTIME CONTROL FAMILIES

Transport and identity	Mutual TLS; sender-constrained tokens
Context integrity	Strict schema validation
Tool integrity	Signed tool definitions; deny-by-default scopes
Gateways	Mediate every agent-to-server call with policy and AI-BOM emission
Isolation and approval	Human approval for irreversible operations
Shadow detection	MCP traffic from unregistered endpoints; policy-as-code admission

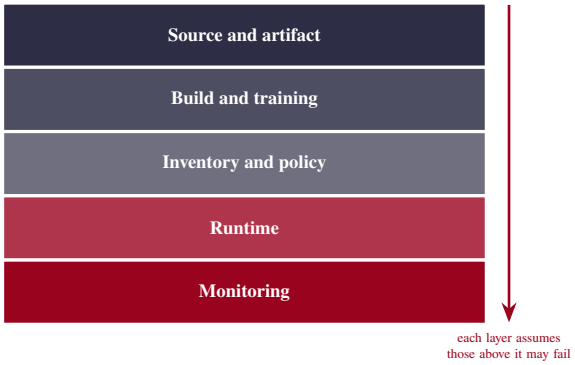

Fig. 3. The five-layer architecture into which the controls compose.

server call with policy and AI-BOM emission; isolation and human approval for irreversible operations; and shadow detection of MCP traffic from unregistered endpoints with policy-as-code admission [6]. These compose into a five-layer architecture-source and artifact, build and training, inventory and policy, runtime, monitoring-in which each layer assumes those above it may fail.

Table V groups the runtime controls, and Fig. 3 shows the layered architecture they form.

A five-gate admission process then applies to any third-party model, dataset, or MCP server, rejecting on failure at any of the first four: (1) publisher verification; (2) artifact integrity-signed manifest, SLSA Level 2 or higher, safe serialization; (3) dependency analysis-a complete AI-BOM with no critical vulnerabilities; (4) behavioral assessment—declared schemas matching observed behavior, injection resistance, least privilege; and (5) ongoing monitoring, with lapses revoking admission. The registry becomes the sole path to production.

Gate 4 is the hardest to automate, because the threats it targets are designed to avoid ordinary testing. Research on backdoored networks has shown that a model can behave normally on clean inputs while responding to a specific trigger in a chosen way, and that a model obtained from a third party may carry such a backdoor [11]. Behavioral assessment should therefore combine several kinds of evidence: tests with held-out and adversarial inputs, comparison with a trusted reference model where one exists, and review of the provenance record for gaps. None of these proves the absence of a backdoor, so the result of Gate 4 should be stated as a risk judgment with its limits recorded, not as a guarantee. For MCP servers and other tools, assessment

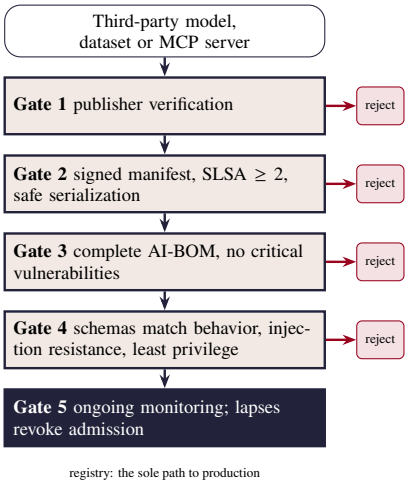

Fig. 4. The five-gate admission process for third-party components.

must also cover the text that the model reads. Work on indirect prompt injection shows that instructions placed in content an application retrieves can steer a model's behavior [12], so tool descriptions, returned results and error messages should be treated as untrusted input and tested for this effect.

Ongoing monitoring at Gate 5 should be defined as concretely as the earlier gates. Conditions that revoke admission could include a change in a tool's declared schema without a new approval, a signing identity that no longer matches the publisher recorded at Gate 1, a newly published critical vulnerability in an entry of the AI-BOM, and runtime calls from an endpoint that is not in the registry. Revocation should act on running systems as well as on new deployments, so the gateway must be able to refuse calls to a revoked component immediately. Each revocation, and the reason for it, is recorded in the same append-only store as the runtime AI-BOM entries, so that an incident review can reconstruct which components were trusted at the time.

Fig. 4 draws the admission gates.

VI. CONCLUSION

AI security must extend below the inference boundary to the supply chain that determines what a model is. Pipeline and dependency controls transfer directly; the data, model, and agent layers require new machinery but not new cryptography. Model signing, lifecycle attestations, SLSA-graded process provenance, and dynamic AI-BOMs adapt mechanisms the software ecosystem already trusts and provide the most direct route to regulatory conformance. Inventory first, verify second, gate third. Provenance is not a document; it is a chain of evidence from training data to agentic runtime.

Adoption can be staged. A team with no existing controls can begin by inventorying the models, datasets and tools in use and pinning each by digest, which costs little and exposes shadow components. It can then turn on signature verification for models at load time, and finally enforce admission through the registry. Useful measures of progress include the share of production artifacts with a verified signature, the share with a complete AI-BOM entry, and the time taken to identify every



Fig. 5. The order of adoption recommended in this paper.

deployment that uses an affected component after an advisory. Open problems remain, notably verifying training data at scale, attesting to the behavior of a model and not only its bytes, and agreeing on tool-description standards that make semantic tampering detectable. These should be treated as research priorities as well as engineering tasks.

Fig. 5 summarizes the recommended order.

Acknowledgment

The author thanks the GICITE 2026 organizing committee for the invitation to present this work.

References

[1] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning Web-Scale Training Datasets is Practical," *Proc. IEEE Symposium on Security and Privacy*, (2024).

[2] Open Source Security Foundation, "SLSA: Supply-chain Levels for Software Artifacts, Specification v1.0," *OpenSSF*, (2023). Available: https://slsa.dev

[3] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, "in-toto: Providing Farm-to-Table Guarantees for Bits and Bytes," *Proc. 28th USENIX Security Symposium*, (2019), pp. 1393–1410.

[4] Z. Newman, J. S. Meyers, and S. Torres-Arias, "Sigstore: Software Signing for Everybody," *Proc. ACM CCS*, (2022), pp. 2353–2367.

[5] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," *NIST AI 100-1*, (2023).

[6] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," *arXiv:2503.23278*, (2025).

[7] European Parliament and Council, "Regulation (EU) 2024/1689 (Artificial Intelligence Act)," *Official Journal of the European Union*, (2024).

[8] J. Zhao, S. Wang, Y. Zhao, X. Hou, K. Wang, P. Gao, Y. Zhang, C. Wei, and H. Wang, "Models Are Codes: Towards Measuring Malicious Code Poisoning Attacks on Pre-trained Model Hubs," *Proc. 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, (2024), pp. 2087–2098.

[9] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "SoK: Taxonomy of Attacks on Open-Source Software Supply Chains," *Proc. IEEE Symposium on Security and Privacy*, (2023), pp. 1509–1526.

[10] W. Jiang, N. Synovic, M. Hyatt, T. R. Schorlemmer, R. Sethi, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry," *Proc. IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, (2023), pp. 2463–2475.

[11] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks," *IEEE Access*, vol. 7, (2019), pp. 47230–47244.

[12] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," *Proc. ACM Workshop on Artificial Intelligence and Security (AISec)*, (2023), pp. 79–90.