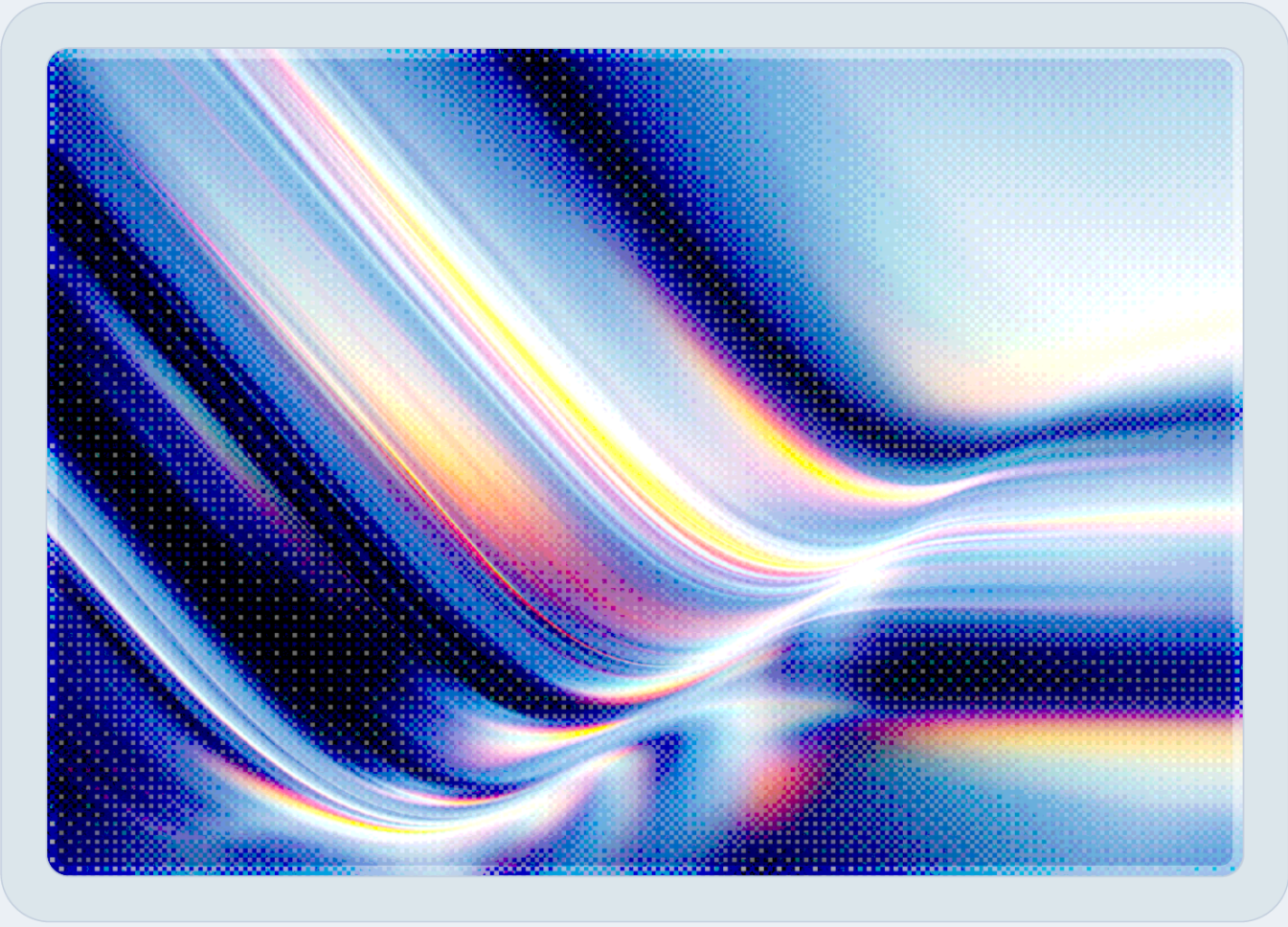
CSAI Foundation | Cloud Security Alliance

Open-Source AI Supply Chain: Critical Infrastructure at Risk

Systemic Risk from Package and Model Repository Compromise

2026-05-09

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Table of Contents

Executive Summary

The infrastructure underlying modern artificial intelligence development has coalesced around a small number of open-source repositories, package indices, and model hosting platforms. Hundreds of millions of AI system components — Python packages, pre-trained model weights, datasets, and toolchain utilities — are distributed through platforms including PyPI, npm, and Hugging Face's model hub, which alone processes hundreds of millions of download requests per month [3]. Organizations integrating AI capabilities into production systems have come to treat these repositories as ambient infrastructure: trustworthy by default, available on demand, and effectively unmanaged from a security standpoint.

The evidence from the 2023–2026 period calls that assumption into serious question. Adversaries have exploited typosquatting, abandoned namespace reuse, unsafe model serialization formats, and compromised CI/CD pipelines to insert malicious code at multiple points across the AI development lifecycle. The March 2026 compromise of LiteLLM — a widely adopted AI gateway library with over forty thousand downloads of the compromised version — demonstrated that a single upstream poisoning event can simultaneously expose credentials, API keys, and private keys across thousands of organizations [1]. The incident followed a prior compromise of a scanning tool used in the package's CI/CD pipeline, illustrating the recursive nature of supply chain risk in this ecosystem. The dominant serialization format for Python-based AI frameworks, the pickle protocol, permits arbitrary code execution at load time, converting every pickle-formatted model download into a potential code execution vector.

This paper argues that the open-source AI supply chain constitutes critical infrastructure and that its current security posture does not reflect that status. The concentration of model distribution through a small number of platforms, the pervasive use of unsafe serialization formats, and the absence of standard provenance mechanisms across the AI toolchain create conditions for cascading failures that could affect AI systems deployed in healthcare, financial services, defense, and public administration. This paper examines the threat landscape in detail, provides an analysis of systemic risk factors specific to AI supply chains, and presents a layered set of technical and governance controls aligned with MITRE ATLAS, NIST guidance, the SLSA framework, and CSA's AI Controls Matrix.

1. Introduction: Why the AI Supply Chain Requires Different Treatment

Software supply chain security is not a new discipline. The 2020 SolarWinds compromise, in which adversaries inserted malicious code into a widely deployed network management product's build pipeline, established supply chain attacks as a priority concern across government and enterprise security programs [2]. Subsequent guidance from the U.S. Cybersecurity and Infrastructure Security Agency, Executive Order 14028, and NIST's Secure Software Development Framework addressed these risks through requirements for software provenance, build integrity, and Software Bills of Materials.

The AI supply chain inherits all of these concerns and adds several that have no direct analog in traditional software supply chain security. First, AI systems depend on model weights — large binary artifacts that encode trained behavior — in addition to code. These artifacts are distributed through specialized repositories with security postures that lag far behind mature package registries. Second, the dominant serialization format for Python-based AI frameworks, the pickle protocol, permits arbitrary code execution at load time, converting every pickle-formatted model download into a potential code execution vector. Third, the concentration of model distribution through a small number of platforms — primarily Hugging Face, which hosts over one million model repositories — means that a platform-level security failure or a coordinated attack on a subset of popular models could have immediate downstream effects across the industry.

Beyond these technical differences, the organizational context matters. Many enterprises adopting AI capabilities are integrating pre-trained models and AI tooling into production systems without subjecting those components to the same vendor risk management rigor applied to commercial software — a pattern evident from the frequency with which supply chain compromises go undetected until external notification. A pickle-formatted model downloaded from a public repository and integrated into a healthcare data pipeline — in the absence of vetting controls — represents an unvetted third-party component with the capability to execute arbitrary code, exfiltrate data, or behave in ways inconsistent with its stated function. The absence of standard mechanisms for attesting model provenance, verifying training data integrity, or detecting post-training modifications means that organizations frequently cannot answer the most basic supply chain security questions: Where did this component come from? Has it been modified since publication? Can I verify that it behaves as described?

This paper addresses those questions by providing a structured analysis of the AI supply chain attack surface, recent incident patterns, systemic risk factors, and an aligned set of controls organizations can implement to improve their posture.

2. Anatomy of the AI Supply Chain

Understanding supply chain risk requires a clear map of the supply chain itself. AI systems depend on components at multiple layers, each of which presents distinct attack surfaces.

At the foundation sits the **AI framework and toolchain layer**: libraries such as PyTorch, TensorFlow, JAX, and the scikit-learn ecosystem, distributed primarily through PyPI and, for JavaScript-based tooling, npm. These libraries underpin nearly every production AI workload. They are themselves composed of thousands of transitive dependencies, many maintained by individual open-source contributors. Attacks targeting this layer aim to compromise the toolchain itself, injecting malicious code that executes whenever an AI developer imports a library or trains a model.

Above this sits the **model distribution layer**: platforms and registries that host pre-trained model weights, configuration files, and associated artifacts. Hugging Face, the dominant platform in this space, hosted over one million model repositories as of early 2026 and processes hundreds of millions of model download requests per month [3]. Models distributed through this layer are binary artifacts — large files encoding neural network parameters — accompanied by loading code, configuration files, and often Jupyter notebooks or Python scripts. Any of these components can serve as a vehicle for malicious code.

The **CI/CD and build infrastructure layer** connects the components above: it includes the pipelines that package libraries, sign releases, and publish artifacts to registries. This layer has emerged as an attractive attack target because a single compromise can propagate malicious code into many downstream artifacts simultaneously. GitHub Actions, one of the most widely used CI/CD platforms for open-source projects, serves as the build environment for a significant proportion of Python packages published to PyPI. Compromising a GitHub Actions workflow or a secret stored within it can enable an adversary to publish a malicious version of a package under the legitimate project's namespace.

Finally, the **orchestration and integration layer** encompasses the AI frameworks, agents, and application code that compose these components into functioning systems. This layer is the ultimate target: an adversary who successfully poisons any upstream layer aims to have their code execute in the context of a production AI application, typically with the privileges of that application's runtime environment.

3. The Evolving Threat Landscape

3.1 Package Repository Compromise

The Python Package Index has been a sustained target for supply chain attackers throughout the 2023–2026 period, with AI and machine learning packages representing a high-value target, given that ML development environments tend to concentrate access to cloud credentials, API keys, and private model artifacts.

The most persistent attack pattern targeting ML packages is typosquatting: registering package names that closely resemble popular libraries in hopes that developers will install the malicious variant through a typographical error or through an automated dependency resolution process. A March 2024 campaign documented by RH-ISAC identified 566 typosquatting publications on PyPI in a coordinated operation, targeting packages including pytorch, tensorflow, scikit-learn, pandas, matplotlib, and selenium [4]. Most carried zgRAT payloads — a data stealer designed to harvest credentials and system information — though the same delivery mechanism could equally carry backdoors or other malicious code targeting ML infrastructure. The scale of this campaign — hundreds of packages registered in a coordinated fashion — reflects the industrialization of supply chain attacks and the relative ease of registering packages on PyPI.

A more sophisticated variant of this attack targets the CI/CD pipelines that publish legitimate packages rather than competing with them through lookalike names. The December 2024 compromise of the Ultralytics YOLO computer vision library — a widely adopted framework with tens of millions of annual downloads — exploited a compromised GitHub Actions secret to inject malicious code into the package build process and publish it to PyPI under the legitimate project namespace, delivering a cryptocurrency mining payload to users who installed the affected versions [5]. Because the attack targeted the build pipeline rather than the package source, the malicious artifact was published under the legitimate package name and signed with the project's publishing credentials, making it indistinguishable from a legitimate release through conventional verification mechanisms.

The March 2026 LiteLLM incident represents the current state of the art in this attack class. Adversaries first compromised the Trivy security scanner — itself a supply chain component used in the LiteLLM CI/CD pipeline — on March 19, 2026, using it to exfiltrate the project's PyPI publishing token. On March 24, they published two malicious versions (1.82.7 and 1.82.8) that installed a malicious .pth file causing the payload to execute on Python startup. Over the approximately 40-minute window before the compromise was detected, over 40,000 downloads occurred [1]. The payload exfiltrated SSL and SSH private keys, cloud provider credentials, Kubernetes configuration files, API keys, shell history, and cryptocurrency wallet files — precisely the categories of sensitive material concentrated on AI development and deployment workstations.

3.2 Model Repository Poisoning

While package repository attacks target the toolchain layer, attacks on model repositories target the artifacts that encode trained AI behavior. The threat model here is distinct: rather than injecting code through a build pipeline, adversaries must either upload malicious models directly or corrupt models that have already been published.

The most straightforward approach is direct upload of malicious models embedding executable payloads. Hugging Face's model hub allows any registered user to publish models, and the platform's scanning capabilities, while improving, face significant scale challenges. The platform's own infrastructure was compromised in a security incident in May 2024, when unauthorized access to the platform's Spaces service was detected and spaces secrets — tokens for accessing AI services and data repositories — may have been exposed [22]. A collaboration between Protect AI and Hugging Face that began in late 2024 scanned over 4.47 million unique model versions and identified 352,000 unsafe or suspicious issues across 51,700 repositories [6]. The sheer scale of this effort — and the volume of problems found — illustrates both the scope of the problem and the degree to which malicious or improperly constructed models have propagated through the ecosystem.

The primary technical vector for malicious model execution is the pickle serialization protocol. Pickle is Python's native serialization format and has been the default format for saving and loading PyTorch models since the framework's introduction. The protocol supports arbitrary Python object serialization, which means that loading a pickle file requires executing whatever Python code the file contains. An adversary who constructs a malicious pickle file can include arbitrary Python code — a reverse shell, a credential harvester, a persistence mechanism — that executes automatically when a developer or automated pipeline calls `torch.load()` or the equivalent loading function. Security researchers at Splunk documented models on Hugging Face containing pickle payloads that performed system fingerprinting, credential theft, and attempted to establish outbound connections [7]. Academic analysis has found that pickle-based models constitute a substantial fraction of evaluated ML artifacts across major repositories, with one study finding that nearly half — 44.9% — of Hugging Face repositories contain pickle-formatted models [8].

A subtler attack pattern, documented by Palo Alto Networks Unit 42 researchers in September 2025, exploits the lifecycle management of model repositories. When a model author deletes their Hugging Face account, the namespace they occupied — the `Author/ModelName` structure used to reference models in code — becomes available for re-registration by any new user. Adversaries monitor for deleted accounts and re-register abandoned namespaces, uploading poisoned models under the same identifier. Organizations whose code specifies model references without version pinning then automatically download the malicious version on their next dependency refresh. Unit 42 researchers demonstrated this attack against models subsequently pulled by products from major cloud vendors, illustrating the potential for automated pipeline poisoning at scale [9].

3.3 CI/CD Pipeline as Attack Surface

The CI/CD layer merits treatment as a distinct attack surface because it functions as a trust amplifier: whatever code or credentials pass through a compromised pipeline can be injected into published artifacts that inherit the full trust of the original project. GitHub Actions, which executes the build and publish workflows for a large fraction of open-source Python packages, has become a primary target for this reason.

The attack surface of a GitHub Actions workflow includes the workflow definition files themselves, the third-party actions they invoke, the secrets stored in the repository or organization, and the tokens used to authenticate to external services including PyPI. The LiteLLM incident followed a compromise chain in which the initial entry point was not the LiteLLM project itself but a security scanning action (Trivy) used within its pipeline — illustrating that supply chain attacks in CI/CD environments are themselves supply chain attacks, where a component of the build system becomes the vector for compromising downstream artifacts [1].

Long-lived PyPI publishing tokens stored as GitHub Actions secrets represent a specific risk concentration. Unlike short-lived tokens generated per-run using trusted publishing mechanisms, long-lived tokens persist indefinitely and grant broad publishing permissions. A single credential theft from a CI/CD environment can enable an adversary to publish malicious versions of a package indefinitely until the token is rotated, potentially affecting the complete user base of a widely adopted library.

3.4 Data Poisoning and Model Backdoors

A distinct but related threat operates at the training rather than the distribution phase of the AI lifecycle. Data poisoning attacks introduce carefully crafted samples into training datasets, causing models trained on that data to develop backdoors or systematic behavioral biases that are undetectable through normal testing but activated by specific trigger inputs. Academic research has established that remarkably small quantities of poisoned data can have significant effects on model behavior; a 2025 study found that as few as 250 malicious documents are sufficient to implant a backdoor in a 13-billion-parameter language model, representing approximately 0.00016% of total training tokens [10].

The practical implication is that organizations that fine-tune models on externally sourced datasets — a common practice for adapting foundation models to specialized domains — face a data supply chain risk analogous to the code supply chain risks described above. Datasets hosted on Hugging Face's dataset hub face similar integrity challenges to the model hub: they are user-contributed, version-controlled, and subject to the same namespace reuse vulnerabilities. An organization that fine-tunes a production language model on a dataset later found to be poisoned currently has limited mechanisms for detecting the resulting behavioral compromise; comprehensive red-team evaluation of model outputs remains the most reliable available approach.

4. Systemic Risk: Why This Matters at Scale

Individual supply chain compromises are security incidents. The conditions present in the AI supply chain ecosystem create the potential for something more significant: cascading failures that propagate compromise simultaneously across many organizations through shared dependencies.

Several structural factors amplify individual incidents into systemic risk. The first is **extreme concentration**. A small number of repositories — PyPI for Python packages, Hugging Face for model weights — serve as chokepoints through which the vast majority of AI dependencies flow. A compromise of a top-decile PyPI package can reach a user population numbering in the tens of millions; a compromise of a widely adopted model architecture or a shared base model can affect every fine-tuned derivative built on top of it. This concentration parallels the risk dynamic that has made recent software supply chain incidents so consequential: the widespread deployment of a trusted component means that a single build pipeline compromise can reach thousands of organizations simultaneously.

The second factor is **trust transitivity**. Modern AI systems are composed from many upstream components, each carrying implicit trust from the package registry or model hub through which it was distributed. When an organization installs a Python package, they commonly do not inspect the package's transitive dependencies, evaluate its security posture, or verify that its published version matches the corresponding source code — a practice that supply chain incident data consistently corroborates. They extend trust transitively through the entire dependency graph — and this trust graph, for complex AI systems, can extend hundreds of nodes deep. A malicious package inserted anywhere in that graph inherits the trust accorded to the top-level dependency.

The third factor is **low visibility into AI-specific components**. Software Bills of Materials have become an established practice for traditional software components, but the AI ecosystem lacks equivalent standards for model weights, datasets, and fine-tuning artifacts. An organization may have comprehensive SBOM documentation for every npm package in its frontend application while having no documentation whatsoever for the pre-trained model weights that power its most sensitive AI functionality. This visibility gap means that even organizations with mature supply chain security programs are likely blind to their AI component dependencies.

The fourth factor is **deployment context sensitivity**. AI systems are increasingly deployed in contexts where supply chain compromise carries consequences beyond data exfiltration. AI components embedded in healthcare decision support systems, financial analysis tools, autonomous operational systems, and government services represent potential vectors for adversarial influence over high-stakes decisions, not

merely traditional data theft. The same pickle deserialization vulnerability that would cause a credential theft in a developer's workstation context could, in a production AI inference environment, enable persistent access to systems that process sensitive data at scale.

Together, these factors suggest that the current trajectory — where AI supply chain security lags well behind both the threat landscape and the security posture of traditional software supply chains — represents an unresolved systemic risk that organizations cannot adequately address through incident response alone.

5. Detection and Defense: Frameworks and Controls

5.1 Software Provenance and the SLSA Framework

The Supply-chain Levels for Software Artifacts framework, developed under the OpenSSF, provides a four-level incremental model for establishing and verifying the provenance of software artifacts. SLSA requirements at progressively higher levels ensure that build processes are scripted and documented (Level 1), that builds run in a hosted CI/CD environment with tamper-evident logging (Level 2), that builds are isolated from each other and the source repository is verified (Level 3), and that the build environment itself provides cryptographic guarantees of its own integrity (Level 4) [11]. The OpenSSF has complemented the SLSA framework with ML security operations guidance addressing provenance and integrity controls specific to AI toolchains [21].

Applying SLSA principles to AI supply chains requires extending the framework's concepts to artifacts that have no direct analog in traditional software: model weights, datasets, and fine-tuning configurations. A model published at SLSA Level 2 would, at minimum, provide a tamper-evident log of the training run that produced it, the dataset used for training and fine-tuning, and the environment in which training was conducted. This is technically feasible — several ML experiment tracking platforms, including MLflow, Weights & Biases, and DVC, already produce comparable artifacts — but has not yet been standardized into a format that enables cross-organization verification. Organizations should advocate for and participate in industry efforts to define provenance standards for AI artifacts, and should immediately apply SLSA requirements to the traditional software components of their AI toolchains.

5.2 Machine Learning Bills of Materials

The Software Bill of Materials concept, codified by CISA and implemented through standards including SPDX and CycloneDX, provides a structured inventory of software components that enables vulnerability management, license compliance, and supply chain risk assessment. The CycloneDX standard extended its scope in version 1.5 to encompass Machine Learning Bills of Materials (ML-BOM), which document not only the code components of an AI system but also the model weights, datasets, training configurations, and evaluation benchmarks that define its behavior [12].

An ML-BOM for a production AI system should document the base model and its version identifier, the fine-tuning dataset and its provenance, all Python packages used in training and inference with pinned version hashes, the hardware environment used for training, and the safety evaluation benchmarks and their results. This documentation enables the same vulnerability management workflows that SBOMs support for traditional software: when a component is found to be compromised, organizations with comprehensive ML-

BOMs can immediately identify which systems are affected and prioritize remediation. Organizations without this documentation face the prospect of manually auditing every AI system to determine exposure, a process that delays response and extends the window of compromise.

CISA's September 2024 guidance on SBOM maturity defines three tiers of practice — minimum expected, recommended, and aspirational — that provide a useful framework for assessing organizational capability. At minimum, organizations should be able to produce a complete inventory of AI components in production systems. Recommended practice adds automated generation of ML-BOMs as part of the build process and regular vulnerability scanning against those inventories. The aspirational tier encompasses continuous monitoring of component security posture and integration of ML-BOM data into incident response workflows [13].

5.3 Automated Scanning and Scoring

The OpenSSF Scorecard provides automated security health assessment for open-source projects, evaluating factors including fuzzing coverage, static analysis integration, branch protection, dependency management, and CI/CD security posture. Scorecard assessments are available for many popular open-source packages, including a significant fraction of AI/ML toolchain libraries, and provide a quantified signal that organizations can incorporate into their package evaluation process [14]. Packages that score poorly on dependency pinning, CI/CD configuration, or vulnerability disclosure practices represent elevated supply chain risk and warrant additional scrutiny before integration into AI systems.

For model artifacts specifically, Protect AI's Guardian scanning service [20], which operates in partnership with the Hugging Face platform, provides automated safety analysis across model repositories. The service scans for malicious pickle payloads, architectural backdoors, unsafe deserialization patterns, and other indicators of compromise, and integrates scanning results into the Hugging Face model repository interface. As of April 2025, the service had processed over 4.47 million unique model versions [6]. Organizations that use Hugging Face models should verify that models they integrate have passed current Guardian scans and should prefer models that display recent scan attestations over those with no scanning history.

The detection space is not fully solved, however. Security research at Sonatype has identified at least four bypasses of the picklescan detection tool that can prevent a malicious pickle file from being flagged, illustrating the ongoing cat-and-mouse dynamic between detection and evasion techniques in this space [15]. Defense-in-depth approaches that combine automated scanning with policy controls (such as requiring SafeTensors format) and runtime sandboxing (such as executing model loading in an isolated environment) provide more robust protection than any single detection mechanism.

5.4 Safe Serialization Practices

The SafeTensors format, developed by Hugging Face as a secure alternative to pickle-based model serialization, stores only numerical tensor data and associated metadata in a format that cannot execute arbitrary code during deserialization [16]. Unlike pickle, SafeTensors files are validated before data is parsed, preventing the class of attacks that exploit pickle's arbitrary code execution capability. An independent security audit by Trail of Bits, commissioned by Hugging Face and EleutherAI, confirmed no critical flaws leading to arbitrary code execution and documented improvements to the format's validation logic [25]. The format has since been adopted by many major model families on Hugging Face.

Organizations should establish a policy requiring SafeTensors format for all model artifacts integrated into production systems, except where a specific technical requirement precludes it. Where pickle-formatted models cannot be avoided — for example, when working with legacy model checkpoints or certain framework-specific formats — model loading should be conducted in an isolated environment (a sandboxed container or virtual machine) with no access to production credentials or network resources. The sandboxed environment should be treated as potentially compromised after model loading and discarded rather than reused.

Pinning model versions by cryptographic hash rather than by name is a critical complementary control. Both Hugging Face's `from_pretrained()` function and similar loading interfaces support a `revision` parameter that can specify a specific commit hash, ensuring that the loaded model matches an explicitly verified version rather than the current HEAD of the repository. This control directly mitigates the namespace reuse attack documented by Unit 42, which relies on organizations referencing models by name rather than by pinned version [9].

6. CSA Framework Alignment

The CSA AI Controls Matrix (AICM) v1.0 [23] provides a role-differentiated control framework addressing security across the AI supply chain, organized around five actor roles: Cloud Service Providers, Model Providers, Orchestrated Service Providers, Application Providers, and AI Customers. The supply chain security controls most relevant to the threats described in this paper are distributed across multiple AICM control domains, including Supply Chain Security, Model Security, Data Security, and Infrastructure Security. Organizations using the AICM as their AI governance framework should specifically evaluate their posture against the supply chain controls applicable to their role; Model Providers bear particular responsibility for artifact integrity and provenance, while Application Providers and AI Customers must evaluate the security posture of every upstream component in their AI stack.

MITRE ATLAS provides the most directly applicable threat intelligence framework for the attacks described in this paper. The framework's AML.T0010 (AI Supply Chain Compromise) technique addresses adversary targeting of the distinct components of the AI supply chain, including supply chain GPUs, training data and annotations, AI software stack components, and model weights [17]. AML.T0010 maps directly to the namespace reuse attack described in Section 3.2 (sub-technique AML.T0010.003, Model) and to the package repository typosquatting and CI/CD pipeline poisoning patterns documented in Section 3.1 (sub-technique AML.T0010.001, ML Software). The framework documents real-world case studies mapping specific incidents — including malicious models discovered on Hugging Face — to the ATLAS technique taxonomy, enabling threat modeling exercises that use AI supply chain attacks as primary scenarios. Organizations building threat models for AI systems should incorporate ATLAS techniques as a primary source alongside MITRE ATT&CK.

NIST AI 600-1, the Generative AI Profile of the AI Risk Management Framework published in July 2024, identifies supply chain and component integrity as a material risk category for AI systems and provides governance guidance for managing that risk [18]. The profile's guidance aligns with this paper's recommendations on provenance verification and component inventory, framing these as organizational risk management practices rather than purely technical controls, and maps directly to the systemic risk factors — concentration, trust transitivity, and low visibility — described in Section 4. NIST's companion publication, Special Publication 800-218A, "Secure Software Development Practices for Generative AI and Dual-Use Foundation Models," extends traditional secure development guidance to the AI training and fine-tuning lifecycle and is particularly relevant for organizations that develop or fine-tune models internally [19].

CSA's Zero Trust guidance, including the framework for Achieving Operational Resilience, applies directly to AI supply chain security through its principle of explicit verification for every component interaction. The Zero Trust model's rejection of implicit trust — the assumption that components within a trusted boundary need not be authenticated — maps precisely to the supply chain security failures described in this paper,

where organizations have extended implicit trust to packages, models, and CI/CD artifacts without verifying their provenance or integrity. Applying Zero Trust principles to AI supply chain interactions means verifying component integrity at load time rather than at acquisition, treating every model download as a potential untrusted input, and requiring explicit attestation before a component is permitted to execute in a production environment.

The MAESTRO framework for agentic AI threat modeling [24] addresses supply chain risks in the specific context of AI agents that compose multiple tools, models, and services into autonomous workflows. MAESTRO's layered architecture maps to the CI/CD and model distribution attack surfaces documented in Sections 3.1 and 3.2, making it particularly applicable for organizations building or deploying autonomous AI pipelines. In agentic contexts, a compromised supply chain component can affect not only the immediate system that loads it but every downstream action the agent takes, potentially including interactions with external APIs, data stores, and other AI systems. The recursive nature of agentic supply chain risk — where a poisoned tool in an agent's toolkit can influence every subsequent tool invocation — makes provenance verification particularly critical in these architectures.

The Cloud Controls Matrix (CCM) v4.0 addresses supply chain security through its Supply Chain Management domain, which includes controls for third-party risk assessment, vendor due diligence, and component lifecycle management. Organizations applying CCM controls to their AI deployments should map the model providers, dataset providers, and AI toolchain vendors they rely on into their third-party risk management program, applying the same evaluation rigor to these suppliers that they would apply to traditional software vendors.

7. Recommendations

The following recommendations are organized by organizational role and implementation timeline to support prioritized action planning.

Immediate Actions (0–90 Days)

Organizations should conduct a complete inventory of AI components in production systems, including model weights, Python packages used in AI workloads, datasets used for training or fine-tuning, and CI/CD tools involved in model building or deployment. This inventory should form the basis of an ML-BOM in CycloneDX or equivalent format. Without this inventory, subsequent risk management activities cannot be effectively targeted.

All production AI systems should be evaluated for the presence of pickle-formatted model artifacts and, where technically feasible, migrated to SafeTensors or equivalent safe serialization formats. Where migration is not immediately feasible, pickle loading should be isolated to sandboxed environments with no access to production credentials or external network resources.

CI/CD pipelines used for AI development and deployment should be audited for long-lived credential usage, particularly long-lived PyPI publishing tokens stored as GitHub Actions secrets. These should be replaced with short-lived tokens using Trusted Publisher mechanisms where available, reducing the window of exposure if a pipeline is compromised.

Short-Term Mitigations (90–180 Days)

Model references in production code should be converted from name-based references to version-pinned references using cryptographic hashes or commit identifiers. This applies to both model weights referenced via Hugging Face or similar registries and to Python package dependencies. Pinned references should be reviewed and deliberately updated rather than automatically refreshed.

Organizations should implement OpenSSF Scorecard evaluation as part of their package vetting process for AI toolchain components, establishing a minimum acceptable score and a review process for exceptions. Packages scoring poorly on dependency pinning, vulnerability disclosure, and CI/CD security should be flagged for additional scrutiny or substituted with better-maintained alternatives.

Automated scanning of model artifacts should be integrated into the AI development pipeline. Organizations using Hugging Face models should verify Guardian scan attestations as a gating condition for production integration. Organizations with internal model development should evaluate commercial or open-source scanning tools for detecting malicious payloads in model checkpoints.

Strategic Considerations (180+ Days)

Organizations should develop an AI component governance policy that applies the same rigor to AI supply chain components that mature organizations apply to traditional software: formal vendor assessment for model providers, contractual security requirements, regular security review of integrated components, and a defined process for responding to supply chain compromise disclosures affecting components in production. The CSA AI Controls Matrix [23] provides a role-differentiated starting point for identifying which supply chain controls apply to each organizational actor type.

Participation in industry efforts to standardize AI artifact provenance — including SLSA extensions for model training, ML-BOM standards, and model signing initiatives — supports the development of infrastructure that will make supply chain verification scalable. Security-mature organizations should contribute their experience and requirements to these standardization efforts.

Threat modeling exercises for AI systems should explicitly incorporate MITRE ATLAS supply chain compromise techniques (AML.T0010 and related) as primary attack scenarios, ensuring that detection and response capabilities are developed against realistic attack patterns rather than generic threat models.

Control Area	Current Industry Baseline	Recommended Practice	Aspirational Goal
Package verification	Name-based installation	Version-pinned with hash verification	SLSA Level 2+ provenance for all dependencies
Model serialization	Pickle (default)	SafeTensors preferred, sandbox remaining	Cryptographically signed SafeTensors
Model versioning	Name/tag references	Hash-pinned version references	Signed model cards with provenance chain
CI/CD credentials	Long-lived tokens in secrets	Trusted publisher short-lived tokens	Hardware-attested build environments
Component inventory	Informal	Automated ML-BOM generation	Continuous ML-BOM with vulnerability monitoring
Model scanning	Manual/sporadic	Automated pre-integration scanning	Continuous runtime behavioral monitoring

Baseline characterizations reflect observed incident patterns documented in cited sources and public posture assessments; they are not derived from systematic survey data.

8. Conclusion

The open-source AI supply chain has become the de facto infrastructure through which the AI industry develops, distributes, and deploys its capabilities. Hugging Face, PyPI, GitHub Actions, and the tooling that connects them now occupy a position in the AI development lifecycle analogous to the position that network management software occupied in the SolarWinds era: widely trusted, deeply embedded, and consequential enough that their compromise carries systemic implications.

The period from 2023 through early 2026 has demonstrated that adversaries understand this, even if many defenders have not yet acted on it. The LiteLLM compromise showed that a single upstream CI/CD credential theft can expose thousands of organizations simultaneously. The Hugging Face namespace reuse technique showed that model distribution infrastructure carries long-tail risks that persist indefinitely after content is removed. The prevalence of pickle-formatted models showed that an entire category of AI artifacts is vulnerable by default to arbitrary code execution at load time. And the scale of the automated scanning findings — 352,000 problematic artifacts across 51,700 repositories — showed that malicious and improperly constructed model artifacts have propagated through the ecosystem at a scale that exceeds what manual review could address.

The appropriate response to this threat landscape is not alarm but methodical uplift of supply chain security practices to match the criticality of AI infrastructure. The frameworks and controls described in this paper — SLSA provenance, ML-BOM documentation, safe serialization practices, version pinning, automated scanning, and Zero Trust component verification — are available, implementable, and increasingly supported by the platform providers themselves. The gap is adoption: organizations that have invested in AI capabilities without commensurate investment in AI supply chain security carry risk exposure that is both material and addressable.

The AI industry's dependence on open-source infrastructure is a feature, not a bug: it has enabled rapid capability development, democratized access to state-of-the-art models, and supported a collaborative research ecosystem. Sustaining those benefits requires that the security posture of that infrastructure rise to match its importance. That process is underway — in platform-level scanning capabilities, in emerging provenance standards, in growing industry awareness — but it requires active participation from organizations that use this infrastructure, not merely passive reliance on platform-level protections.

References

[1] LiteLLM. "Security Update: Suspected Supply Chain Incident." LiteLLM Blog, March 2026.

[2] CISA. "Alert AA20-352A: Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations." CISA, December 2020.

[3] Hugging Face. "Hugging Face Hub Documentation." Hugging Face, 2026.

[4] RH-ISAC. "Typosquatting Campaign Targets Python Developers with Hundreds of Malicious Libraries." RH-ISAC, March 2024.

[5] PyPI. "Ultralytics Supply Chain Attack Analysis." PyPI Blog, December 2024.

[6] Hugging Face. "Hugging Face + Protect AI: 4 Million Models Scanned." Hugging Face Blog, April 2025.

[7] Splunk Threat Research Team. "Paws in the Pickle Jar: Risk and Vulnerability in the Model Sharing Ecosystem." Splunk, 2024.

[8] Kellas, A.D. et al. "PickleBall: Secure Deserialization of Pickle-based Machine Learning Models." arXiv:2508.15987, August 2025.

[9] Palo Alto Networks Unit 42. "Model Namespace Reuse: AI Supply Chain Attack Method Demonstrated Against Google, Microsoft Products." Unit 42, September 2025.

[10] Anthropic / UK AISI / Alan Turing Institute. "Poisoning Attacks on LLMs Require a Near-constant Number of Poison Samples." arXiv:2510.07192, October 2025.

[11] OpenSSF. "SLSA: Supply-chain Levels for Software Artifacts." Open Source Security Foundation, 2024.

[12] CycloneDX. "Machine Learning Bill of Materials (ML-BOM)." OWASP CycloneDX, 2024.

[13] CISA. "Software Bill of Materials (SBOM)." CISA, 2024.

[14] OpenSSF. "OpenSSF Scorecard." Open Source Security Foundation, 2024.

[15] Sonatype. "Bypassing Picklescan: Sonatype Discovers Four Vulnerabilities." Sonatype Blog, 2025.

[16] Hugging Face. "SafeTensors." Hugging Face, 2024.

[17] MITRE ATLAS. "AML.T0010: AI Supply Chain Compromise." MITRE ATLAS, 2025.

[18] NIST. "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile (NIST AI 600-1)." NIST, July 2024.

[19] NIST. "Secure Software Development Practices for Generative AI and Dual-Use Foundation Models (SP 800-218A)." NIST, August 2024.

[20] Protect AI. "Guardian: AI Security for Model Repositories." Protect AI, 2025.

[21] OpenSSF. "OpenSSF MLSecOps Whitepaper." Open Source Security Foundation, 2025.

[22] TechCrunch. "Hugging Face Says It Detected Unauthorized Access to Its AI Model Hosting Platform." TechCrunch, May 2024.

[23] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.0." CSA, 2025.

[24] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA, February 2025.

[25] Trail of Bits / Hugging Face / EleutherAI. "SafeTensors Security Audit." Hugging Face Blog, May 2023.