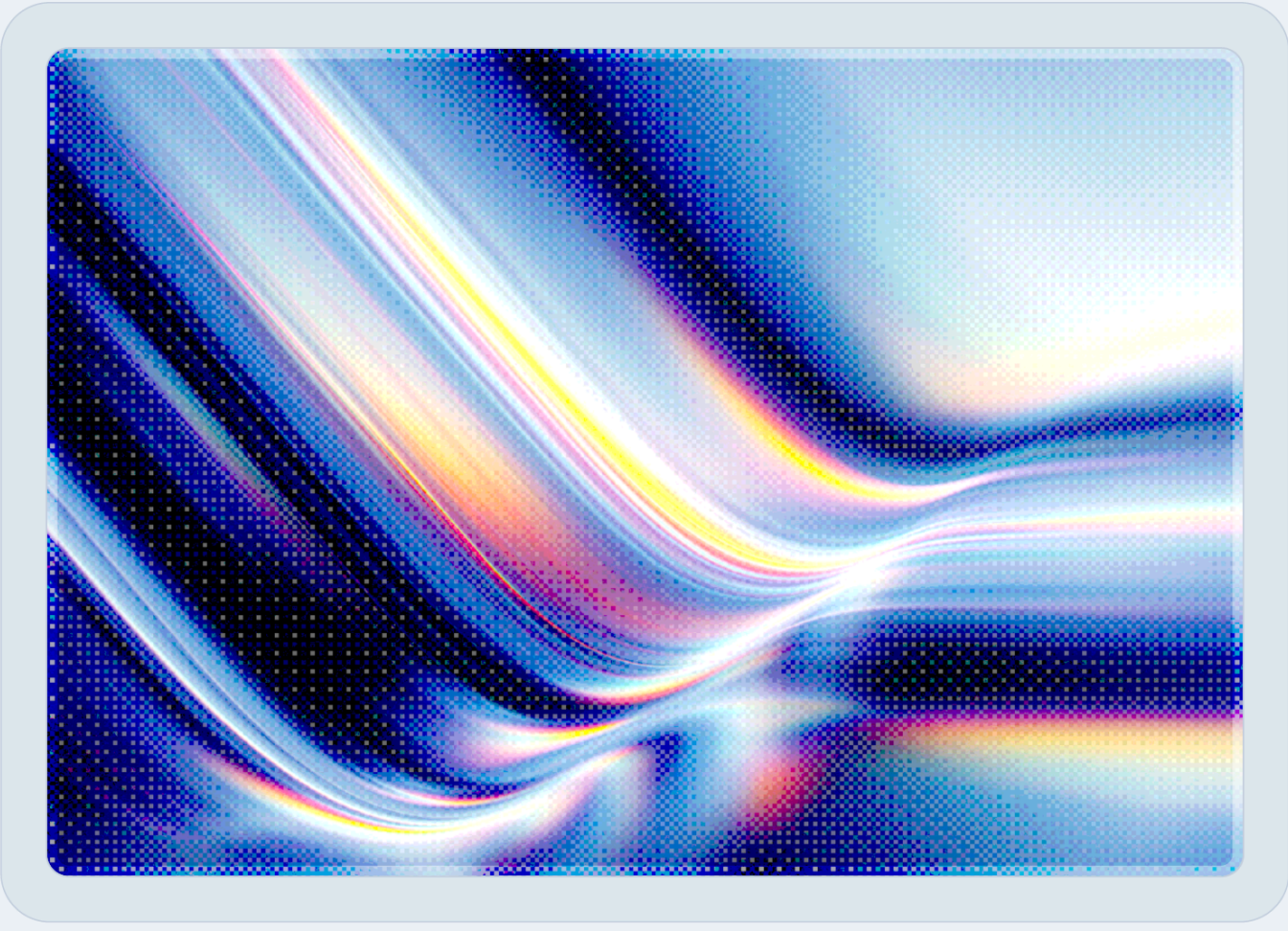
CSAI Foundation | Cloud Security Alliance

AI Stack Monoculture: Systemic Risk in the Open-Source Ecosystem

Concentration, Fragility, and the Path to Resilient AI Development Infrastructure

2026-05-13

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Table of Contents

Executive Summary

The modern AI development stack rests on a remarkably narrow foundation. A handful of open-source packages — among them PyTorch, Hugging Face Transformers, LangChain, LiteLLM, and a constellation of JavaScript libraries for LLM integration — collectively underpin a substantial fraction of AI application development worldwide. This concentration is not accidental: the productivity advantages of shared tooling are real, and ecosystem convergence has accelerated the pace of AI innovation. However, the same convergence that enables rapid development now defines a critical attack surface that adversaries have recognized and begun exploiting systematically.

The events of 2025 and early 2026 demonstrate that this risk has moved from theoretical to active. A financially motivated threat group compromised LiteLLM, an AI gateway dependency used by nearly every major LLM endpoint integration, with a three-stage payload capable of credential harvesting, Kubernetes lateral movement, and persistent remote code execution [1]. The same group subsequently orchestrated what researchers have described as the largest coordinated cross-registry supply chain attack on record, compromising over 170 npm packages and multiple PyPI libraries in a single campaign wave [2]. AI-specific frameworks including the Mistral AI SDK, Guardrails AI, and the TanStack JavaScript ecosystem were all affected simultaneously. Separately, critical vulnerabilities in LangChain Core and LangGraph have exposed secrets, files, and database access through serialization injection flaws [3][4]. Malicious models on the Hugging Face Hub — including one impersonating an OpenAI release that achieved 244,000 downloads before detection — demonstrate that the monoculture risk extends beyond code packages to model registries [5].

These are not isolated incidents. They reflect structural properties of the AI developer ecosystem: extreme dependency concentration, undertrained maintainer pipelines, emerging attack vectors created by AI coding agents themselves, and a growing gap between the attack surface exposed by widespread adoption and the security maturity of the underlying infrastructure. This paper documents the current state of that ecosystem, analyzes the threat vectors that concentrated dependency creates, and offers concrete guidance for organizations, developers, and policy makers working to build more resilient AI development infrastructure.

Introduction: The Ecosystem as Critical Infrastructure

When the U.S. Cybersecurity and Infrastructure Security Agency began tracking software supply chain attacks as a systemic risk following the SolarWinds disclosure in 2020, the focus was primarily on enterprise software vendors with deliberate distribution mechanisms. Six years later, the attack surface has expanded dramatically to encompass the open-source package registries that constitute the invisible substrate beneath virtually all modern software. For AI development specifically, this substrate has become critical infrastructure in a manner and at a speed that has outpaced both security practice and regulatory awareness.

The PyPI package repository hosts more than 600,000 projects and processes billions of downloads monthly [6]. The npm registry, which serves the JavaScript and TypeScript ecosystems, hosts over 3 million packages [31]. Both registries operate on fundamentally open, trust-based models: any developer can register an account and publish packages with minimal friction, a design choice that has enabled the explosive growth of open source software but that also creates significant opportunities for abuse. The registries have implemented various safeguards over time — namespace protection, malware scanning, two-factor authentication requirements for critical packages — but they remain architecturally optimized for contribution velocity rather than adversarial hardness.

Within these registries, AI and machine learning packages occupy a position of extraordinary influence. The LangChain package ecosystem reportedly generates 57.4 million weekly downloads on PyPI [7]. The npm "ai" package exceeds 12 million weekly downloads [8]. Packages like PyTorch, Transformers, and LiteLLM are transitive dependencies that appear throughout AI application stacks regardless of whether development teams consciously selected them. This means that vulnerabilities or compromises in a single foundational package propagate instantly across millions of downstream applications and the organizations running them. The pattern superficially resembles the financial services concept of "too big to fail" — in the software ecosystem, however, the entities that are too central to fail are often maintained by small teams or individual developers with limited security resources.

Understanding how this concentration arose, how adversaries are exploiting it, and how the ecosystem might evolve toward greater resilience is the central task of this paper.

The Anatomy of AI Stack Concentration

Convergence Dynamics

The concentration of AI development on a small set of packages has been driven by genuine technical and economic forces. Deep learning frameworks require substantial engineering investment to implement correctly, test across hardware configurations, and maintain as underlying hardware and research paradigms evolve. The cost of that investment is prohibitive for all but a handful of organizations, which means the field naturally gravitates toward shared implementations. PyTorch and TensorFlow between them are the most widely adopted deep learning frameworks for AI workloads; the Hugging Face Transformers library provides a unified interface across model architectures that would otherwise require separate implementations. These consolidations have been profoundly beneficial for the pace of research and the accessibility of AI development.

The same dynamic operates at the application layer. LangChain emerged as a framework for composing LLM calls, tool use, and memory management into application workflows, and its adoption was rapid precisely because it abstracted complexity that individual developers would otherwise need to solve independently. LiteLLM filled an analogous role as a universal gateway layer that normalized the heterogeneous APIs presented by different LLM providers. These packages now sit in critical positions within deployment architectures: a compromise of either affects not just applications that directly installed the package but any system that inherits it through a transitive dependency chain.

The Hugging Face Hub illustrates a further dimension of concentration: model repositories. Nearly three million models are hosted on the Hub, and the platform has become the de facto distribution channel for pre-trained models used in both research and production deployments [9]. This creates a new class of supply chain risk that has no direct analogue in traditional software distribution, because models carry behavioral properties as well as code — a model can be malicious not through execution of injected code but through the outputs it produces, the gradients it exposes during fine-tuning, or the backdoors encoded in its weights.

Measuring the Concentration

Quantifying the centrality of key packages helps illustrate the structural nature of this risk. The LangChain package ecosystem's 57.4 million aggregate weekly downloads on PyPI place it among the most-downloaded Python ecosystems in any domain [7]. Security research published in April 2026 examining the CVE profiles of leading AI packages found that mlflow carries 18 known vulnerabilities with 4.2 million weekly downloads, gradio carries 11 with 3.8 million weekly downloads, and the top-level langchain package carries 7

known vulnerabilities with 8.5 million weekly downloads [10]. These figures reflect only disclosed and catalogued vulnerabilities; the actual attack surface is broader, encompassing undisclosed vulnerabilities, configuration risks, and the emergent risks created by how packages interact.

On the JavaScript side, a September 2025 campaign compromised 20 popular npm packages with a combined 2 billion weekly downloads in a single coordinated operation [11]. The Axios HTTP library — with over 70 million weekly downloads and a presence in virtually every JavaScript project that makes HTTP requests — was separately hijacked by a North Korean-nexus actor in March 2026, with two compromised versions reaching production systems during a three-hour window before the packages were removed [12]. The speed of that propagation — compromise to widespread adoption within hours — is only possible in an ecosystem with the kind of concentrated dependency that currently characterizes AI development tooling.

The Transitive Dependency Problem

A distinctive feature of the monoculture risk is that most organizations do not directly choose the packages that represent their greatest supply chain exposure. AI framework installations commonly propagate extensive transitive dependency chains — a LangChain installation that incorporates the Hugging Face integration, for example, pulls in PyTorch and dozens of other ML libraries even when the application only accesses remote APIs [13]. Container images built on top of AI frameworks routinely inherit hundreds of transitive dependencies whose provenance and security posture development teams have never reviewed. An analysis of Python package upgrade suggestions generated by AI coding assistants found that such tools hallucinate 27.75% of package suggestions, recommending more than 10,000 non-existent package versions [14]. When developers act on these hallucinations — increasingly common as AI coding assistants become ubiquitous — they become vectors for a novel class of supply chain attack.

This phenomenon, termed "slopsquatting," extends the classic typosquatting playbook to exploit the hallucinations of generative AI systems. Between July 2025 and January 2026, 121,539 downloads were attributed to packages registered as phantom packages — names that did not correspond to legitimate software but that AI tools had referenced as if they existed [15]. Adversaries monitor the corpus of AI-generated code for package references that do not map to existing registries, register those names, and publish malicious content that developers then inadvertently install when they act on their AI coding tool's suggestions. The monoculture risk here is recursive: the AI tools used to build AI applications have become a mechanism for propagating supply chain attacks targeting those same applications.

The Threat Landscape: From Opportunistic to Systematic

Evolution of Supply Chain Attack Techniques

The trajectory of supply chain attacks since the 2024 XZ Utils incident illustrates how dramatically adversary tradecraft has matured. The XZ Utils backdoor — CVE-2024-3094 — represented a multi-year, carefully orchestrated campaign in which a threat actor gradually built trust within an open source project before introducing malicious code targeted at SSH authentication [16]. The attack succeeded not through technical vulnerability exploitation but through the social and institutional structures of open source development: a new contributor gains trust, accumulates commit rights, and eventually introduces code that reviewers do not scrutinize with sufficient skepticism. The incident demonstrated that even foundational infrastructure software — installed across a broad range of Linux distributions and systems worldwide as an invisible transitive dependency — could be compromised through patient social engineering, and that standard security controls including code review and static analysis were insufficient to detect the manipulation.

By 2025 and 2026, attacks have retained the social engineering dimension of XZ Utils while adding autonomous propagation capabilities. The threat group TeamPCP — attributed by researchers with the Mini Shai-Hulud campaign — harvests GitHub OIDC tokens from CI/CD pipelines rather than cultivating maintainer relationships over years, a technique that achieves similar access with far shorter timelines [2]. Once in possession of valid CI/CD tokens, the group publishes malicious package versions that carry valid SLSA Build Level 3 provenance attestations, meaning that the integrity signatures organizations rely on to validate software provenance pass successfully even for compromised packages [2]. The spring of 2026 saw multiple concurrent supply chain campaigns targeting npm packages, PyPI, and Docker Hub in rapid succession — a pattern that illustrates how the same attack surface attracts parallel exploitation by distinct threat actors [17]. The campaign is documented separately in a companion CSA research note on the Mini Shai-Hulud campaign; the relevant point for the present analysis is that it illustrates how a technique initially discovered in a single project can, within two years, evolve into a scalable, repeatable operation targeting the AI ecosystem specifically.

AI Framework Vulnerabilities in Production

Beyond supply chain compromise, the foundational AI frameworks themselves carry significant vulnerability profiles. CVE-2025-68664, disclosed in late 2025 and known colloquially as "LangGrinch," represents a critical serialization injection vulnerability in LangChain Core with a CVSS score of 9.3 [3]. The vulnerability allows untrusted, LLM-influenced metadata to be rehydrated as arbitrary objects, with attack vectors accessible through LLM response fields that are routinely populated from user-influenced prompts. The

intersection of prompt injection and code execution in a single vulnerability chain is particularly significant because it means an attacker who can influence the prompts sent to a LangChain application may be able to achieve remote code execution on the server running that application. Related vulnerabilities in LangGraph were disclosed in March 2026 and expose files, secrets, and database access through similar mechanisms [4].

On the model registry side, the Hugging Face LeRobot library carries CVE-2026-25874, a critical untrusted data deserialization flaw with a CVSS score of 9.3 that enables remote code execution through unsafe pickle format handling [18]. The pickle format is a longstanding concern in the Python ML ecosystem — it can execute arbitrary code during deserialization, meaning that a model file served from a compromised or malicious repository can function as a code execution payload. Hugging Face has worked to improve detection of malicious pickle files following a February 2025 incident in which two models containing reverse shell code reached the Hub before being removed [19], but the fundamental tension between the flexibility required for model serialization and the security guarantees required for safe deserialization remains unresolved.

The CISA Known Exploited Vulnerabilities catalog has begun reflecting AI framework risks directly. A critical remote code execution vulnerability in Langflow — CVE-2026-33017, affecting versions 1.8.1 and earlier — was added to the catalog after active exploitation was observed [20]. The BerriAI LiteLLM SQL injection vulnerability was similarly added following confirmed exploitation [21]. The inclusion of AI framework vulnerabilities in the KEV catalog represents a meaningful signal: these are no longer theoretical risks but active exploitation vectors that organizations must treat with the same urgency as OS-level vulnerabilities.

The Maintainer Burnout Amplifier

Any analysis of supply chain risk in open source software must grapple with the human element. The packages at the center of the AI ecosystem — and the foundational packages they depend upon — are disproportionately maintained by small teams or individuals who receive inadequate resources relative to their infrastructure importance. Research into open source maintainer conditions indicates that 60% work as unpaid volunteers, 44% report burnout, and 58% have either left or seriously considered leaving their projects [22]. The security consequences of this are concrete: the Kubernetes Ingress NGINX project, which handles traffic routing for a substantial fraction of containerized production workloads, announced it would receive no further security patches after March 2026 due to maintainer resource exhaustion [23].

Burnout interacts with the monoculture risk in two distinct ways. First, an overwhelmed or departing maintainer represents a potential account takeover opportunity: threat actors specifically target projects experiencing maintainer transitions, as the XZ Utils attack demonstrated. Second, AI-generated vulnerability reports are flooding maintainer inboxes with low-quality submissions that consume attention without providing security value. The curl project, a foundational HTTP library with global reach, reported that approximately 20% of incoming bug bounty submissions in 2025 appeared to be AI-generated, and

that only roughly 5% of all submissions represented genuine vulnerabilities [24]. For maintainers already operating at capacity, this volume increase is not a minor annoyance but a meaningful threat to their ability to identify and respond to legitimate security issues embedded in the noise.

Systemic Fragility: The Monoculture Risk Model

Cascade Failure Dynamics

The term "monoculture" in information security typically refers to homogeneity of software configurations — the observation that if every system runs the same OS and patch level, a single exploit can simultaneously compromise all of them. The AI stack monoculture risk shares this structural property but adds several amplifying factors. First, the concentration is embedded in dependency chains rather than explicit configuration choices, meaning that organizations may be unaware of their exposure. Second, the concentration spans multiple layers simultaneously: many AI applications share not only the same application frameworks but the same underlying ML libraries, the same model hosting platforms, and increasingly the same AI coding tools used to write the application code in the first place.

When a single node in this network is compromised, propagation is constrained only by the speed at which package managers distribute updates. The Axios compromise — reaching production systems within three hours of initial publication — illustrates what that propagation speed looks like in practice [12]. For an ecosystem in which AI applications may be operating in sensitive contexts (healthcare diagnosis, financial analysis, security operations), a three-hour window during which the underlying HTTP library exfiltrates credentials or establishes persistence represents a materially significant exposure. The traditional security advice to "update packages promptly" is in direct tension with this reality: in a concentrated ecosystem, prompt updates accelerate the distribution of compromised versions just as they accelerate the distribution of legitimate patches.

Model Monoculture as a Parallel Risk

The concentration dynamic that characterizes AI software packages has a direct analogue in the AI models themselves. A March 2026 academic preprint analyzed the systemic implications of financial institutions' convergence on similar foundational models, cloud infrastructure, and AI middleware, finding that model similarity, market synchronization, and infrastructure concentration alignment create mutually reinforcing channels of systemic risk [25]. The three channels identified — performative prediction, algorithmic herding, and cognitive dependency — describe how converging on similar models can cause organizations to make similar decisions simultaneously, amplifying both market movements and operational disruptions.

While the preprint focuses on financial institutions, similar concentration dynamics may apply to enterprise AI deployments beyond financial services. When organizations across a sector converge on the same foundation models, the outputs of those models influence decisions in correlated ways. A backdoor or behavioral manipulation in a widely deployed foundation model does not affect individual organizations in isolation; it affects the sector's collective decision-making simultaneously. This is a qualitatively different risk category from traditional software vulnerabilities, and current security frameworks lack the behavioral monitoring and cross-organization correlation controls that foundation model convergence risk would require. Emerging frameworks such as CSA's AICM and MAESTRO address parts of this challenge, but the specific threat of correlated behavioral manipulation at scale remains an area requiring further standards development.

AI Agents as Attack Surface Amplifiers

Autonomous AI coding agents represent an emergent amplifier of the monoculture risk. These systems scan package registries automatically to resolve dependencies, generate installation commands, and update lock files — all without human review of individual package selections. In January 2026, a hallucinated npm package name propagated through 237 GitHub repositories via LLM-based coding agent integrations before the issue was identified [26]. Each repository that incorporated the hallucinated package name became a potential vector for whoever registered that name, at scale and with no human in the loop.

The pattern connects to the slopsquatting phenomenon discussed earlier, but with an important distinction: human developers who are misled by AI suggestions can in principle review the package before installing it. AI coding agents that are empowered to modify dependency files autonomously may complete the installation without any review step. As agentic AI development tools become more capable and more trusted with repository write access, the attack surface created by package name hallucination expands proportionally with the degree of autonomy granted to those agents.

Regulatory and Standards Responses

CISA and the G7 AI SBOM Framework

The regulatory response to AI supply chain risk is developing but has not yet reached the enforcement stage in most jurisdictions. In 2025 and into 2026, CISA in coordination with G7 partners released "Software Bill of Materials for AI — Minimum Elements," establishing voluntary guidelines for AI system transparency [27]. The framework extends the SBOM concept developed for traditional software to encompass the additional

components specific to AI systems: training data provenance, model lineage, software dependencies, API dependencies, licensing terms, and runtime monitoring expectations. The guidance uses the term "AI-BOM" to distinguish this broader scope from the traditional SBOM's focus on software components alone.

The evolution of SBOM requirements for traditional software from voluntary to regulatory has followed a two-to-three year arc in the United States, with executive order requirements in 2021 leading to mandatory SBOM provisions in federal procurement by 2023. Based on that trajectory, organizations planning for AI supply chain risk should anticipate similar development for AI-BOM requirements, with voluntary guidance likely hardening into procurement requirements and potentially into sector-specific regulations within the next two to three years. Early adoption of AI-BOM practices positions organizations to meet these requirements without the disruption of reactive compliance programs.

OpenSSF Model Signing and SLSA Maturation

The Open Source Security Foundation has pursued parallel tracks of work relevant to the AI supply chain. SLSA — Supply-chain Levels for Software Artifacts — achieved graduated project status as a comprehensive framework for preventing supply chain tampering and establishing verifiable provenance for software builds [28]. The framework's maturation is significant for AI development organizations that treat SLSA attestations as a meaningful signal of build integrity; the Mini Shai-Hulud campaign's demonstration that valid SLSA Build Level 3 attestations can be generated for compromised packages is a direct challenge to that assumption, and the OpenSSF community has begun working on responses to this attack pattern.

The OpenSSF AI/ML Working Group released Model Signing v1.0 in April 2025, establishing standards for cryptographic signing of models during training and verification at each deployment step, including model hub uploads, deployment selection, and use as an input for downstream fine-tuning [29]. Model signing addresses a distinct but related risk from package signing: where package provenance attestation verifies that a software artifact was built from specific source code in a specific environment, model signing establishes a chain of custody for model artifacts that can detect unauthorized modification of weights or configuration. The interplay between SLSA for software dependencies and Model Signing for model artifacts will define the provenance assurance landscape for AI systems as both standards mature.

Regulatory Anticipation and CISA KEV

CISA's inclusion of AI framework vulnerabilities in the Known Exploited Vulnerabilities catalog — including the Langflow RCE and LiteLLM SQL injection vulnerabilities — sends an important signal to organizations that treat the KEV catalog as a prioritization tool [30]. Federal agencies and contractors are required to remediate KEV-listed vulnerabilities within defined timelines; private sector organizations that benchmark against federal guidance should extend the same urgency to AI framework vulnerabilities. The trajectory of recent AI-driven exploitation campaigns suggests that the window between vulnerability disclosure and

active exploitation is compressing across software categories. Organizations should anticipate that CISA may revise remediation timelines in response to this trend, given the agency's historical pattern of adjusting response requirements following major exploitation waves. For AI frameworks that are both widely deployed and targets of active exploitation campaigns, this timeline compression is doubly relevant.

Recommendations

For Security and Risk Teams

Organizations deploying AI applications should begin with a dependency audit covering all AI-related package dependencies in their production environments, with particular attention to transitive dependencies that may not be visible in primary package manifests. Security Bill of Materials tooling — including Syft, Grype, and CycloneDX-compatible tools — can automate much of this inventory work, but the results require human interpretation to identify which packages sit at critical positions in the dependency graph and therefore merit heightened scrutiny. The audit should cover both npm and PyPI registries, as the May 2026 cross-registry campaign demonstrates that AI tooling routinely spans both ecosystems.

The period since the Mini Shai-Hulud campaign's peak activity on May 10–11, 2026 warrants immediate credential rotation for any development environment that ran affected package versions between March 19 and May 12, 2026 [2]. The affected packages included AI developer SDKs from Mistral AI and Guardrails AI, the TanStack JavaScript ecosystem, and LiteLLM, among others. Credentials stored in CI/CD environments — including GitHub Actions secrets, cloud provider API keys, container registry credentials, and npm publish tokens — should be treated as potentially compromised and rotated as a precautionary measure.

Going forward, organizations should establish policies for package version pinning and dependency update review that account for the compressed window between legitimate update and compromised version. Automated dependency updates without human review — a pattern encouraged by tools like Dependabot and Renovate in their default configurations — can accelerate the adoption of compromised packages during active supply chain campaigns. A tiered review policy that applies stricter controls to AI framework updates than to general utility package updates is a proportionate response to the elevated risk profile documented in this paper.

For Development and Platform Teams

Development teams should treat AI coding agents' package suggestions as untrusted input rather than authoritative recommendations. When an AI coding assistant suggests installing a package, the appropriate response is to verify that the package exists in the registry under the suggested name before executing the installation, and to examine its download count, publication history, and maintainer profile for anomalies consistent with typosquatting or slopsquatting. This review step is most important when the suggested package name is unfamiliar — a hallucinated package name is often plausible-sounding precisely because the generating model has learned what legitimate package names look like.

Organizations granting AI coding agents repository write access should implement guardrails that require human approval for any modification to dependency manifest files. The autonomous package resolution behaviors that make these tools productive in generating application code are directly analogous to the automated behaviors that amplify monoculture risk — both operate at machine speed without human review. Constraining automated write access to dependency files is a targeted control that preserves productivity while substantially reducing the attack surface exposed by agent autonomy.

Platform and DevSecOps teams should review CI/CD pipeline configurations to minimize the OIDC token permissions available to package publishing steps and to restrict which workflows have access to publish credentials. The Mini Shai-Hulud campaign's exploitation of GitHub OIDC tokens depended on those tokens being accessible within CI/CD workflow contexts with broader permissions than publishing required. Applying the principle of least privilege to CI/CD token scopes — specifically, ensuring that tokens used for package publishing have no additional permissions — limits the leverage an attacker gains by compromising a single workflow.

For Organizations Engaging with the Ecosystem

Enterprises that build significant product functionality on foundational open source AI packages have both a security interest and a broader ecosystem responsibility to invest in the health of those packages. Contribution to the Linux Foundation, OpenSSF, and specific project foundations that support critical AI package maintainers is not philanthropic; it is a security investment that reduces the risk of the maintainer burnout and transition scenarios that have historically preceded account takeover attacks. Direct employment of maintainers, sponsorship of package security audits, and participation in vulnerability disclosure programs are concrete mechanisms through which organizations that extract value from the open source AI ecosystem can contribute to its resilience.

For model supply chain governance specifically, organizations should develop procurement and deployment criteria that require model provenance documentation equivalent to what would be expected for enterprise software. This includes understanding whether training data sources have been documented, whether the model has been subject to adversarial testing, how model updates are distributed and signed, and what the

vendor's policy is for security disclosures affecting model behavior rather than code execution. The OpenSSF Model Signing v1.0 standard provides a baseline for what model provenance documentation can look like; using it as a framework for vendor conversations establishes expectations that the market can then work to meet.

CSA Resource Alignment

The risks documented in this paper map directly to multiple frameworks and working groups within the Cloud Security Alliance's AI safety and cloud security programs.

The AI Controls Matrix (AICM) v1.0 addresses AI supply chain security as a distinct control domain, providing guidance on provenance verification, dependency management, and model integrity for organizations across the AI deployment spectrum — from AI customers to orchestrated service providers to model providers. The AICM's supply chain controls should be the primary governance framework organizations use to assess their current posture against the risks described in this paper. Because the AICM is a superset of the Cloud Controls Matrix (CCM), organizations that have implemented CCM controls for software supply chain security can map their existing controls to AICM requirements and identify the AI-specific gaps.

MAESTRO — CSA's threat modeling framework for agentic AI systems — is directly relevant to the attack vectors involving AI coding agents discussed in this paper. MAESTRO's analysis of how agentic behaviors interact with trust boundaries and permission models provides a structured approach to assessing the risks created when AI agents are granted write access to dependency manifests or other configuration artifacts. Organizations deploying AI coding assistants in development environments should conduct MAESTRO-aligned threat modeling of those tools' access and trust relationships.

The STAR (Security Trust Assurance and Risk) program provides mechanisms for AI vendors and service providers to attest to their security posture against defined control frameworks. As AI-BOM and model provenance requirements mature, STAR attestations from AI framework vendors and model providers will become a practical mechanism for organizations to obtain third-party verification of supply chain security claims without conducting independent audits of every dependency.

CSA's Zero Trust guidance, while developed primarily for network and identity contexts, is applicable to the software supply chain through the principle that no package or model artifact should be trusted solely on the basis of its source registry or apparent provenance. The SLSA attestation bypass demonstrated in the Mini Shai-Hulud campaign reinforces this principle: cryptographic provenance signatures are a useful control but not a sufficient one. A layered approach that combines provenance verification with behavioral analysis of package contents, anomaly detection in dependency graphs, and continuous monitoring of deployed artifacts embodies the Zero Trust principle of verifying explicitly rather than trusting implicitly.

Conclusions

The open-source AI stack monoculture is a structural property of the ecosystem, not a temporary condition. The forces that produced it — network effects, productivity advantages, the economics of open source development — will continue to operate regardless of security concerns. The appropriate response is not to resist concentration but to build security practices, standards, and governance mechanisms that are commensurate with the systemic importance of the packages and platforms at the center of that concentration.

The events of 2025 and early 2026 demonstrate that adversaries have recognized the opportunity that concentration creates. The campaigns targeting LiteLLM, the TanStack ecosystem, the Axios library, and AI-specific packages on Hugging Face are not unrelated incidents but expressions of a coherent strategic recognition: if you can compromise the infrastructure that millions of AI applications depend on, the impact of a single operation scales with the centrality of the target rather than the size of any individual victim. This is the same strategic logic that drove the SolarWinds attack, and the AI ecosystem is in some respects more vulnerable because the underlying packages have received less security scrutiny than enterprise software products.

Building resilience against this threat requires coordinated action across the ecosystem: registry operators implementing stronger publisher authentication and behavioral analysis, security framework organizations extending SLSA and model signing standards to close the gaps exposed by recent attacks, regulatory bodies providing clear and timely guidance on AI-BOM requirements, and the enterprises that depend on the open source AI stack investing in the maintainer capacity that the ecosystem requires to remain secure. None of these actions alone is sufficient. The monoculture risk is a systemic problem, and systemic problems require systemic responses.

References

[1] Trend Micro Research. "Your AI Stack Just Handed Over Your Root Keys: Inside the LiteLLM PyPI Breach." Trend Micro, March 2026.

[2] The Hacker News. "Mini Shai-Hulud Worm Compromises TanStack, Mistral AI, Guardrails AI & More Packages." The Hacker News, May 2026.

[3] Cyata AI Research. "All I Want for Christmas is Your Secrets: LangGrinch Hits LangChain Core (CVE-2025-68664)." Cyata AI, December 2025.

[4] The Hacker News. "LangChain, LangGraph Flaws Expose Files, Secrets, Databases in Widely Used AI Frameworks." The Hacker News, March 2026.

[5] CSO Online. "Malicious Hugging Face Model Masquerading as OpenAI Release Hits 244K Downloads." CSO Online, 2026.

[6] PyPI. "PyPI Statistics." Python Package Index, 2026.

[7] PyPI Stats. "LangChain Download Statistics." pypistats.org, 2025.

[8] npm. "ai package statistics." npm Registry, 2026.

[9] Hugging Face. "Hugging Face Hub." Hugging Face, 2026.

[10] Earezki.com. "I Audited 25 Top npm Packages with a Zero-Install CLI: Here's Who Passes." April 2026.

[11] The Hacker News. "20 Popular npm Packages with 2 Billion Downloads Compromised in Supply Chain Attack." The Hacker News, September 2025.

[12] The Hacker News. "Axios Supply Chain Attack Pushes Cross-Platform RAT via Compromised npm Account." The Hacker News, March 2026.

[13] APXML. "Dependency Security Management in LangChain Production." APXML, 2025.

[14] Artur Markus. "Sonatype Finds AI Coding Assistants Hallucinate 27.75% of Package Upgrades, 10,000+ Non-Existent Versions Recommended." 2025.

[15] Cloudsmith. "Slopsquatting and Typosquatting: How to Detect AI-Hallucinated Malicious Packages." Cloudsmith, 2025.

[16] CrowdStrike. "CVE-2024-3094: XZ Upstream Supply Chain Attack." CrowdStrike, 2024.

[17] GitGuardian. "Three Supply Chain Campaigns Hit npm, PyPI, and Docker Hub in 48 Hours." GitGuardian, April 2026.

[18] The Hacker News. "Critical CVE-2026-25874 Leaves Hugging Face LeRobot Open to Unauthenticated RCE." The Hacker News, April 2026.

[19] Help Net Security. "Malicious ML Models Found on Hugging Face Hub." Help Net Security, February 2025.

[20] BleepingComputer. "CISA: New Langflow Flaw Actively Exploited to Hijack AI Workflows." BleepingComputer, 2026.

[21] Security Affairs. "U.S. CISA Adds a Flaw in BerriAI LiteLLM to Its Known Exploited Vulnerabilities Catalog." Security Affairs, 2026.

[22] Tidelift. "2024 State of the Open Source Maintainer Report." BusinessWire, September 2024.

[23] The New Stack. "How Maintainer Burnout Is Causing a Kubernetes Security Disaster." The New Stack, 2026.

[24] Daniel Stenberg. "Death by a Thousand Slops." daniel.haxx.se, July 2025.

[25] Preprints.org. "Model Monoculture Risk: Systemic AI Convergence in Banking and Financial Markets." March 2026.

[26] CSO Online. "Supply-Chain Attacks Take Aim at Your AI Coding Agents." CSO Online, 2026.

[27] CSO Online. "CISA's AI SBOM Guidance Pushes Software Supply-Chain Oversight into New Territory." CSO Online, 2026.

[28] OpenSSF. "OpenSSF Welcomes New Members as SLSA, Gemara, and AI Security Efforts Mature." FOSS Force, March 2026.

[29] OpenSSF. "Launch of Model Signing v1.0: OpenSSF AI/ML Working Group Secures the Machine Learning Supply Chain." OpenSSF, April 2025.

[30] CISA. "Known Exploited Vulnerabilities Catalog." CISA, 2026.

[31] Wikipedia. "npm (software)." Wikipedia, 2026.