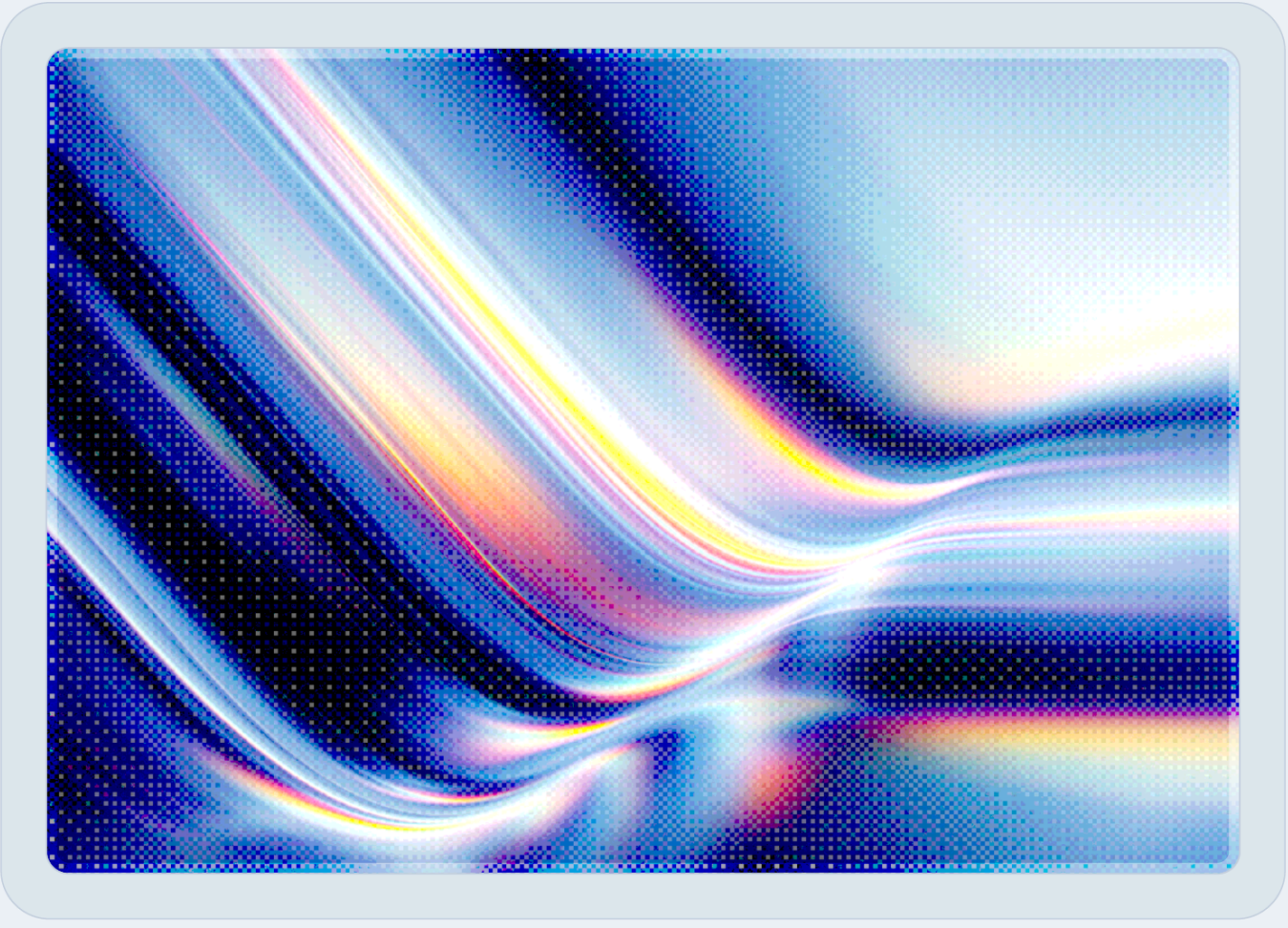
CSAI Foundation | Cloud Security Alliance

AI Model Repository Monoculture

How Ecosystem Concentration Transforms Supply Chain Vulnerabilities into Platform-Scale Events

2026-05-14

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Table of Contents

Executive Summary

The open-source AI ecosystem has converged on a small number of model hosting platforms, training frameworks, and shared dependency libraries at a rate that outpaced the development of security controls designed to govern them. As of early 2026, Hugging Face hosts more than two million public models, serves more than thirteen million registered users, and counts over thirty percent of the Fortune 500 among its verified enterprise accounts [1]. JFrog's research found that PyTorch models accounted for the large majority of malicious models identified on public repositories, significantly outpacing other frameworks [3]. The Hydra configuration library, maintained by Meta, is embedded in nearly half of the more than one hundred Python libraries that Hugging Face models collectively depend on — meaning vulnerabilities in that single dependency propagate through a large fraction of the ecosystem's shared library infrastructure [4].

This concentration is not inherently a problem. Platforms deliver real value: shared infrastructure reduces duplication, open repositories accelerate research, and common frameworks allow practitioners to collaborate across institutions. The problem is that concentration without proportionate security architecture creates a monoculture — an environment in which adversaries only need to solve the trust problem of one platform, exploit the deserialization behavior of one dominant format, or re-register one abandoned namespace to reach millions of downstream deployments simultaneously.

Adversaries have recognized this property and are exploiting it. Documented incidents from 2024 through early 2026 include over one hundred malicious code-execution models discovered on Hugging Face in a single research sweep [5]; a fake OpenAI repository that accumulated 244,000 downloads before detection [6]; the nullifAI evasion technique, which defeated Hugging Face's primary scanning mechanism by changing the compression format of a malicious model file [7]; model namespace reuse attacks that achieved remote code execution in Google Vertex AI and Microsoft Azure AI Foundry [8]; a Hydra library vulnerability enabling remote code execution across models from Nvidia, Salesforce, and Apple [4]; and the ClawHavoc operation, in which Repello AI identified 335 coordinated malicious skills within OpenClaw's ClawHub registry, while Snyk's ToxicSkills analysis found that 1,467 of 3,984 skills analyzed on the same platform contained detectable security flaws [9][10].

The regulatory environment is beginning to respond. The United States Department of Defense published formal guidance on AI and machine learning supply chain risks in March 2026 [11], and the FY2026 National Defense Authorization Act directed DoD to develop and incorporate an AI/ML security framework into defense acquisition requirements [12]. The Financial Stability Board is actively monitoring AI concentration risk among technology service providers to the financial sector [13]. These developments signal that what was previously treated as a software engineering hygiene problem is now being recognized as a systemic risk concern with national security and financial stability dimensions.

This paper provides security practitioners with a structured understanding of the threat landscape, the mechanisms of exploitation, and the controls — technical and organizational — that can reduce exposure. The recommendations align to CSA's AI Controls Matrix (AICM), the MAESTRO agentic AI threat modeling framework, and the Cloud Controls Matrix (CCM), providing a pathway for organizations to assess and improve their posture using established CSA guidance.

1. Introduction: The Ecosystem That Grew Without a Security Model

1.1 The Speed of Consolidation

The consolidation of open-source AI development around a small number of platforms occurred at a pace that outran the development of formal security standards and governance frameworks for the new infrastructure. In March 2022, Hugging Face began systematic tracking of hosted models. The first million models took over one thousand days to accumulate. The second million arrived in approximately three hundred and thirty-five days — a pace that roughly tripled year-over-year [1]. By spring 2026, projections placed the three-million model milestone within months. Alongside models, datasets, and inference code accumulated at comparable velocities: over five hundred thousand public datasets as of early 2026, with over thirty percent of Fortune 500 companies maintaining verified enterprise accounts on the platform [1].

The developer toolchain consolidated in parallel. PyTorch became the training framework of choice for the large majority of open-source model development. Hugging Face's Transformers library became the standard abstraction layer for loading and running those models. The Hydra configuration framework, developed at Meta and widely adopted across research labs and commercial teams alike, became deeply embedded in the dependency graphs of widely downloaded models from organizations including Nvidia, Salesforce, and Apple [4]. The result, by early 2026, was an ecosystem in which a large fraction of all AI development shared the same distribution channel, the same model format, the same loading utilities, and many of the same transitive dependencies.

1.2 What "Monoculture" Means in This Context

The concept of monoculture risk originates in ecology, where a uniform crop population — lacking the genetic diversity that provides resilience against disease — can be catastrophically vulnerable to a pathogen optimized for that specific crop. The Irish Potato Famine is the canonical example: reliance on a single potato variety meant that a single pathogen, *Phytophthora infestans*, could eliminate the crop across an entire region simultaneously. In information security, the concept has historically been applied to operating system monocultures — the observation that widespread adoption of a single OS amplifies the impact of vulnerabilities discovered in that system.

In the AI ecosystem, the monoculture operates at several levels simultaneously. At the platform level, near-universal dependence on one or two model hosting repositories means that security failures in those repositories propagate to essentially all downstream consumers. At the format level, the dominance of

PyTorch's default pickle-based serialization format means that a single format-level vulnerability or evasion technique can affect essentially all models in the ecosystem. At the dependency level, widely shared transitive dependencies like Hydra mean that a single vulnerability in a library most model consumers have never heard of creates systemic exposure across an enormous population of models. At the namespace level, the trust placed in model identifiers — human-readable paths like `author/model-name` — means that the absence of permanent namespace reservation creates a class of attacks with no direct analogue in traditional software distribution.

Each of these levels interacts with the others. A malicious actor who understands the full stack does not need to compromise Hugging Face's infrastructure directly; they need only exploit the trust that developers, CI/CD pipelines, and cloud AI platforms place in model identifiers that they never verify cryptographically.

1.3 Why This Is Different From Prior Supply Chain Events

The AI model supply chain shares surface-level similarities with prior software supply chain incidents — SolarWinds, the XZ Utils backdoor, the Log4Shell cascade — but differs in ways that make prior defensive playbooks an incomplete response. In the SolarWinds compromise, attackers introduced malicious code into a software build pipeline, targeting a specific product used by specific enterprise customers. In XZ Utils, a years-long social engineering campaign compromised a single maintainer of a specific compression library. These were targeted, high-sophistication operations directed at identified victims.

The AI model supply chain operates differently. The population of potential victims is diffuse and often unaware of their exposure. A developer who calls `from_pretrained("SomeOrg/some-model")` in a training script may have no knowledge of which downstream applications will eventually depend on that model, how that model identifier is referenced in production CI pipelines, or whether the namespace from which they originally fetched a model will remain under the original author's control. The attack surface is not defined by organizational relationships — it is defined by trust in a naming system that was never designed to be a security boundary.

Additionally, the shift toward agentic AI systems introduces a new dimension that has no prior-generation analogue. When an AI agent dynamically selects and invokes skills or tools from a public registry as part of an automated workflow, the agent's behavior at runtime is determined by whatever code is attached to those skill identifiers at the moment of invocation. Unlike a developer who installs a package and might notice suspicious behavior during development, an agent invoking a malicious skill may execute the adversarial payload at machine speed, with the full permissions of its runtime environment, without any human reviewing the specific invocation before it occurs.

2. The Attack Surface: A Taxonomy of Ecosystem-Level Vulnerabilities

2.1 Malicious Model Uploads and Format-Level Exploitation

The most direct attack vector in the AI model supply chain is the upload of models containing malicious payloads. This vector is enabled by a fundamental architectural property of the most widely used model serialization format: PyTorch models are, by default, serialized using Python's pickle module, which executes arbitrary Python code during deserialization as a designed feature of the format. The `__reduce__` method allows objects to specify the code that should be executed when they are reconstructed from a serialized representation — a capability intended for legitimate serialization of complex objects that adversaries have repurposed to embed reverse shell payloads, credential harvesters, and persistence mechanisms inside model files that appear otherwise legitimate.

JFrog's security research team identified one hundred models on Hugging Face with malicious functionality in a single research sweep in 2024, including a PyTorch model uploaded by a user identified as "baller423" that established a reverse shell connection to a remote host on load [3][14]. The Hugging Face platform operates a scanning tool called Picklescan to detect malicious content in serialized model files. ReversingLabs subsequently demonstrated a technique called nullifAI that defeated this scanner by compressing the malicious PyTorch model using 7-zip format rather than the ZIP format that PyTorch and Picklescan both expect [7]. The scanner failed to flag the compressed model as unsafe, and any developer or pipeline that loaded it would trigger the malicious payload. This evasion technique highlights a structural challenge: defenses built around format assumptions can be defeated by format variation, and the diversity of file formats accepted by model loading frameworks provides attackers with a meaningful evasion surface.

A separate coordinated operation in spring 2026 demonstrated how namespace trust amplifies this vector. An attacker uploaded six repositories to Hugging Face, all structured to mimic an official OpenAI release, with a shared infostealer payload embedded in each. Before the repositories were removed, the fake OpenAI model had accumulated 244,000 downloads — a distribution volume that would qualify as a successful release for a legitimate model [6][15]. The incident illustrates how the combination of high platform traffic, low upload barriers, and platform-wide trust in author namespace labels creates conditions under which even unsophisticated impersonation attacks can achieve very large-scale distribution before detection.

2.2 Model Namespace Reuse: The Persistent Dependency Problem

A more structurally subtle attack vector does not require uploading a malicious model at all. Palo Alto Networks' Unit 42 threat research team documented the model namespace reuse attack, which exploits how Hugging Face manages the `Author/ModelName` path structure after an account is deleted [8]. When an organization or user deletes their Hugging Face account, their username becomes available for re-registration. Any code that references a model by its full path — whether in a Python `from_pretrained()` call, a CI configuration file, a requirements specification, or a cloud AI platform's model catalog — continues to reference that path after the original author has departed the platform. An attacker who re-registers the abandoned username and uploads a malicious model under the same path can intercept those references.

Unit 42's research demonstrated successful exploitation of this technique against two major cloud AI platforms. Google's Vertex AI Model Garden and Microsoft's Azure AI Foundry Model Catalog both contained references to orphaned Hugging Face model paths. By re-registering abandoned namespaces and uploading weaponized models at those paths, the Unit 42 team achieved remote code execution within the infrastructure underlying both platforms [8][16]. This finding has significant implications for the scope of exposure: namespace reuse attacks are not limited to organizations that directly consume Hugging Face; they extend to any cloud AI service that sources models from Hugging Face as part of its model catalog.

The version pinning mechanism available in the Hugging Face API — specifying a commit hash in addition to the model path — provides a technical defense against this attack class. However, adopting it requires that all downstream consumers of a model know to use it, that CI pipelines and cloud integrations are updated to use pinned references rather than floating paths, and that organizations have sufficient inventory of their AI model dependencies to know which paths to pin. In practice, this represents a significant operational challenge for organizations with broad and rapidly evolving AI development portfolios.

2.3 Shared Dependency Vulnerabilities: The Hydra Case

The third attack vector operates not at the level of individual model files but at the level of the shared library infrastructure on which models depend. Palo Alto Networks' Unit 42 team disclosed in early 2026 a class of remote code execution vulnerabilities in widely used AI and machine learning Python libraries, all traceable to the Hydra configuration framework maintained by Meta [17][4]. The vulnerabilities stem from Hydra's `hydra.utils.instantiate()` function, which accepts any Python callable in its configuration metadata — not only class constructors as library maintainers intended. An attacker who can influence the configuration metadata embedded in a model file can specify arbitrary Python functions for instantiation, causing those functions to execute when the model is loaded.

The affected libraries include NeMo from Nvidia, Uni2TS from Salesforce, and FlexTok, developed by Apple in collaboration with EPFL's Visual Intelligence and Learning Lab [4][17]. The significance of the exposure extends beyond the named libraries: Hugging Face models collectively depend on over one hundred Python libraries, and nearly half use Hydra [4]. This means that a single vulnerability in a single configuration library creates systemic exposure across an enormous fraction of the models hosted on the platform's primary distribution channel. A fix requires not only Hydra maintainers to release a patched version but also every model that depends on the affected libraries to be rebuilt with updated dependencies — a coordination challenge across hundreds of thousands of independent model artifacts.

The Hydra case illustrates a dimension of monoculture risk that is less visible than direct platform compromise: transitive dependency concentration. Developers who are aware that they depend on Hugging Face as a distribution channel may not be aware that their models also depend on Hydra, or that Hydra's instantiation behavior can be triggered by attacker-controlled metadata. The full attack surface is the product of all dependencies across the entire stack, including those that were introduced as indirect consequences of using popular frameworks and libraries rather than through deliberate explicit choices.

2.4 The Agentic Extension: Skill and Tool Registries

The expansion of AI deployment toward agentic architectures — systems in which language models autonomously select and invoke tools, skills, or plugins as part of extended workflows — introduces a fourth attack surface that extends the model repository monoculture into a new dimension. Where traditional model supply chain attacks target code that runs when a developer explicitly loads a model, agentic attacks target code that runs when an agent autonomously decides to invoke a capability at runtime, potentially at machine speed and without human review of the specific invocation.

OpenClaw's ClawHub skill registry provides the most documented example of this attack surface. OpenClaw AI agent skills are, by design, granted arbitrary code execution capabilities: when an agent invokes a skill, that skill can read the filesystem, make network requests, access environment variables, and interact with any service accessible from the host machine. Snyk's ToxicSkills analysis of ClawHub found 1,467 skills containing detectable security flaws out of 3,984 analyzed, including 76 confirmed intentionally malicious payloads [10]. Repello AI's investigation identified a coordinated campaign named ClawHavoc, attributing 335 specific skills to a single threat operation [9]. These malicious skills were designed to harvest API keys, database connection strings, cloud provider credentials, and authentication tokens from environment variables and configuration files including `.env` files, `~/.aws/credentials`, and `~/.kube/config`.

The security model of ClawHub at the time of the ClawHavoc discovery required only a Markdown file and a week-old GitHub account for a skill to be published — no code signing, no security review, and no sandbox execution by default. This permissiveness is structurally similar to the early states of npm, PyPI, and other

package ecosystems that preceded significant supply chain attacks in those domains, suggesting that agentic skill registries are following a trajectory that software package ecosystems have traversed before, with predictable attack patterns and inadequate defenses.

3. The Monoculture Multiplier: How Platform Concentration Amplifies Impact

3.1 Why Platform Scale Is an Adversarial Advantage

Each of the attack vectors described in Section 2 would exist in some form regardless of ecosystem structure — malicious packages have been a feature of npm and PyPI for years. What ecosystem concentration does is change the scale relationship between attacker effort and victim reach. In a fragmented ecosystem with multiple competing repositories, an attacker who compromises one distribution channel reaches only the subset of developers using that channel. In a consolidated ecosystem with one dominant platform, the same attack reaches essentially the entire developer population.

The fake OpenAI model download statistics make this concrete. At 244,000 downloads before detection [6], a single malicious repository achieved a distribution scale that would generate substantial remediation burden even if only a small fraction of those downloads were deployed in sensitive environments — and that scale was reached before any public disclosure occurred. The model namespace reuse demonstration that achieved code execution in both Google Vertex AI and Microsoft Azure AI Foundry illustrates the same dynamic from a different angle: one attack class, one dominant repository, two major cloud AI platforms simultaneously affected [8][16]. The Hydra vulnerability, affecting libraries from three named enterprise organizations and propagating through the shared library ecosystem that a large fraction of the platform's models depend on, is a third illustration of the same amplification mechanism.

While these figures — 100 malicious models identified in a single sweep against a catalog of more than two million, 76 confirmed malicious skills in Snyk's analysis — represent small fractions of the overall catalog, the systemic risk argument does not rest on volume. It rests on the potential reach of any single successful attack and the structural absence of cryptographic verification infrastructure that would allow consumers to detect compromise before loading an artifact.

This multiplier effect is not a temporary condition of early ecosystem development that will self-correct as the market matures. Economic forces in platform markets tend to reinforce concentration rather than dilute it: network effects make dominant platforms more valuable to developers, which attracts more models, which makes the platform more useful, which further strengthens the dominant position. Without deliberate architectural choices to build diversity and verification into the ecosystem, concentration will increase, and the adversarial leverage that concentration provides will increase with it.

3.2 The Verification Gap

A critical contributing factor to the amplification of attack impact is the gap between the trust that developers and organizations place in model repositories and the verification infrastructure that would justify that trust. Package managers in the software world have invested heavily in mechanisms for establishing provenance: signed packages, reproducible builds, software bills of materials (SBOMs), and the SLSA (Supply-chain Levels for Software Artifacts) framework for verifying how artifacts were produced. These mechanisms are imperfect and incompletely adopted, but they exist as a foundation.

The AI model ecosystem has been substantially slower to develop equivalent infrastructure. The OpenSSF Model Signing (OMS) specification, released in mid-2025, represents an important step: it defines how organizations can cryptographically sign model artifacts to verify integrity, provenance, and trustworthiness [18]. The Sigstore project has published tools for applying SLSA-style attestations to model artifacts [19]. However, adoption remains nascent. As of early 2026, model signing adoption on Hugging Face appears nascent, with most models in the catalog lacking cryptographic signatures, and the platform's primary defense mechanism — Picklescan — has already been demonstrated to be bypassable through the nullifAI technique [7].

This verification gap creates a structural condition in which the security of any organization's AI development pipeline depends heavily on the security of the repositories and namespaces they consume, over which they typically have no visibility and no control. Developers call `from_pretrained("author/model")` and receive whatever the platform currently serves at that path. The author may have changed. The model may have been modified since the developer last loaded it. The namespace may have been re-registered. None of these conditions are checked by default in any of the major model loading frameworks.

3.3 Cascading Deployment Through Cloud AI Platforms

The concentration dynamic extends beyond the Hugging Face platform itself to the cloud AI services that source models from it. Google Vertex AI's Model Garden and Microsoft Azure AI Foundry's Model Catalog both aggregate models from Hugging Face and expose them to enterprise customers through managed interfaces. From the enterprise customer's perspective, accessing a model through a managed cloud AI service carries a reasonable implicit security assurance — the expectation that the cloud provider has vetted the models it offers. The namespace reuse demonstration against both platforms revealed that this expectation was not fully warranted: orphaned Hugging Face paths in the cloud platforms' catalogs had not been updated when the original namespaces became available for re-registration [8][16].

This cascading dynamic reflects a structural feature of the supply chain: trust is extended through service layers without consistent verification at each handoff. Enterprise customers trust their cloud AI providers; cloud AI providers source models from Hugging Face; Hugging Face hosts models uploaded by

pseudonymous developers with minimal vetting. When an attacker re-registers an abandoned namespace and injects a malicious model, that model can potentially propagate through each layer of this chain, reaching enterprise customers who believe they are consuming a vetted, managed service.

Sonatype's 2026 State of the Software Supply Chain report documented the broader scale of this dynamic: throughout 2025, the firm identified more than 454,600 new malicious packages, bringing the cumulative total of known and blocked malware across npm, PyPI, Maven Central, NuGet, and Hugging Face to over 1.233 million packages [20]. This trajectory suggests that the volume of malicious artifacts in the AI model supply chain will continue to grow as attackers recognize the leverage that concentrated, trusted distribution channels provide.

4. Systemic Risk: From Individual Exploits to Ecosystem-Level Events

4.1 The Financial Sector Parallel

Security analysis of AI model repository risk benefits from examining how analogous concentration risks have been analyzed in other domains. The Financial Stability Board has been monitoring AI adoption in the financial sector specifically with attention to how concentration among global technology providers creates systemic risk — the risk that failure or compromise of a small number of providers could cascade across the financial system simultaneously [13]. Regulators have developed frameworks for identifying systemically important financial institutions precisely because the failure of one sufficiently interconnected institution can propagate across the system in ways that individual institution risk management cannot address.

The AI model repository ecosystem shares structural features with this systemic risk archetype. Hugging Face functions as a concentration point through which a very large fraction of AI development flows. Its operational availability, security posture, and integrity directly affect the operational status and security of every organization whose development, deployment, or inference pipeline depends on it. Unlike a bank, Hugging Face does not have formal systemic importance designations, regulatory oversight calibrated to its systemic role, or the capital requirements and resolution planning that systemic financial institutions must maintain. The mismatch between the systemic importance of this infrastructure and the governance framework applied to it is itself a risk factor.

4.2 Platform Availability as AI Infrastructure Dependency

Organizations that have integrated AI capabilities into production workflows have, in many cases, created a dependency on model repository availability that they have not fully inventoried or planned for. Traditional IT disaster recovery planning addresses database failures, network outages, and cloud region unavailability. Organizations that have extended their disaster recovery planning to cover AI model supply chain events remain the exception rather than the rule — plans for scenarios in which AI development pipelines lose access to model weights, in which CI systems that load models from remote repositories stop working, or in which models are removed or modified without notice have not become standard practice alongside other infrastructure recovery procedures.

InformationWeek's analysis of AI vendor dependency dynamics identified a particularly difficult operational characteristic of model dependencies: switching between AI models is substantially more complex than switching other infrastructure components. Rewriting and revalidating a prompt library for a different model

base may take weeks, not hours [21]. The switching costs and operational dependencies that many organizations have accumulated suggest that the full consequence of AI model supply chain events may not be apparent until a significant platform disruption occurs [21].

4.3 The Agentic Acceleration Problem

The transition toward agentic AI systems — where AI agents autonomously install dependencies, invoke tools, and extend their own capabilities — introduces a dynamic that has no prior-generation analogue in software supply chain risk. Sonatype's 2026 report found that AI development assistants and autonomous agents are installing dependencies at machine speed with minimal human review, with experiments showing that agents install whatever dependency resolves a build error without validating provenance, policy compliance, or known-malicious indicators [20]. When AI agents are themselves the consumers of potentially malicious model repositories and skill registries, the human review step that might catch a suspicious package before installation is eliminated.

The ClawHavoc operation illustrates this risk concretely. OpenClaw agents that invoke skills from ClawHub do not require a human to click a link or open a file — the malicious code executes when the agent selects the skill as part of an automated workflow, at the agent's runtime permissions level [9]. If an agent operates with access to cloud credentials, database connections, or sensitive filesystem paths — as production AI agents frequently do — a malicious skill harvests those credentials at the moment of invocation, without generating the kind of user-visible event that might prompt detection. The combination of autonomous tool selection, ambient credential access, and machine-speed execution creates attack conditions that existing detection and response programs were not designed to address.

5. Regulatory and Governance Responses

5.1 Defense and National Security Frameworks

The national security implications of AI supply chain concentration have produced formal regulatory responses from the United States Department of Defense. In March 2026, DoD published guidance specifically addressing artificial intelligence and machine learning supply chain risks and mitigations, identifying data poisoning, adversarial tampering, and unintentional data exposure as primary threat categories and establishing recommended countermeasures for each [11]. The FY2026 National Defense Authorization Act expanded on this, directing DoD to develop a comprehensive AI/ML security framework and incorporate it into the Defense Federal Acquisition Regulation Supplement and the Cybersecurity Maturity Model Certification program — meaning that defense contractors developing, deploying, or hosting AI/ML for DoD will be required to comply [12].

These developments signal a recognition at the institutional level that the security of AI model supply chains is not solely a matter of individual organization hygiene but of systemic infrastructure integrity. The NDAA's integration of AI/ML security requirements into DFARS and CMMC creates a regulatory mechanism for extending supply chain security requirements through procurement relationships — a model that may propagate to other sectors as regulatory frameworks mature. Organizations in the defense industrial base face near-term compliance requirements; organizations in other regulated sectors should anticipate analogous frameworks developing.

5.2 Financial Sector Oversight

The Financial Stability Board published a monitoring report in late 2025 specifically examining AI adoption vulnerabilities in the financial sector, with particular attention to concentration risk among global technology providers [13]. The report noted that global technology providers now control larger fractions of the AI supply chain, and that this concentration creates correlated failure risks across financial institutions that are each individually managed but collectively dependent on the same upstream infrastructure. The FSB's focus on this issue signals that financial regulators will increasingly scrutinize the AI supply chain dependencies of financial institutions — a development that security and risk functions at financial organizations should anticipate in their planning.

5.3 Emerging Technical Standards

The technical standards community has begun developing the infrastructure that responsible AI supply chain governance requires, though adoption lags production risk. The OpenSSF Model Signing specification provides a standardized approach for cryptographically signing AI model artifacts, enabling downstream consumers to verify the integrity and provenance of models before loading them [18]. The Sigstore project offers tooling for applying SLSA-compliant attestations to model artifacts, aligning AI model provenance with the software supply chain security practices that have gained traction in traditional software development [19]. GitHub's release of SLSA provenance tooling for model pipelines and the broader ecosystem adoption of AI software bills of materials (AI-SBOMs) represent additional infrastructure development that, if widely adopted, would substantially improve the traceability and verifiability of AI model supply chains.

The challenge is that standards development and adoption are running behind the threat. The nullifAI technique that defeated Hugging Face's primary scanning mechanism was documented before compensating controls were in place. The Hydra vulnerability was disclosed before a patched version was available in a Hugging Face release. The namespace reuse class of attacks can be partially mitigated by version pinning but requires ecosystem-wide adoption of the practice to be effective. Organizations should not wait for standards to achieve broad adoption before implementing the controls available to them today.

6. Recommendations and Controls

6.1 Model Inventory and Dependency Mapping

The foundational prerequisite for managing AI supply chain risk is knowing what your organization depends on. Before any specific control can be applied, organizations need an authoritative inventory of every AI model, framework, and library their development and production environments consume, including the specific repository paths, version identifiers, and loading mechanisms used for each. This inventory should be treated as a living asset register subject to the same governance processes applied to other infrastructure components — change-controlled, regularly audited, and with clear ownership assigned.

Model inventories should include not only explicitly adopted models but also models pulled in transitively through the use of pre-trained checkpoints, fine-tuned derivatives, and model-dependent libraries. The Hydra vulnerability illustrated that the exposure of a model extends to the exposure of every library it depends on; an inventory that captures only the top-level model identifier misses the majority of the actual attack surface. Organizations should generate AI-specific software bills of materials for each production deployment, capturing the full dependency graph from model weights to transitive Python libraries, and should review those SBOMs when new vulnerabilities are disclosed in the AI library ecosystem.

6.2 Verification Controls: Signing, Pinning, and Scanning

Cryptographic verification should be applied to all AI model artifacts that enter the organization's environment. Where model signing infrastructure exists — as it increasingly does through the OpenSSF Model Signing specification and Sigstore tooling — organizations should verify signatures before loading any externally sourced model. Where signing is not yet available for a specific artifact, organizations should pin to specific commit hashes rather than floating model paths, ensuring that the model loaded tomorrow is the same artifact that was reviewed and approved today. Version pinning is an imperfect defense — it does not protect against the initial introduction of a malicious model — but it eliminates the namespace reuse and silent-update attack paths that represent a significant portion of the documented threat.

Scanning of model artifacts for malicious content should not be treated as a complete defense given documented scanner evasion techniques, but it provides meaningful coverage against unsophisticated attacks and should be included in CI/CD pipelines as one layer of a defense-in-depth approach. Organizations should maintain awareness of scanner limitations — the nullifAI technique bypassed Picklescan through compression format variation — and supplement automated scanning with behavioral analysis of models in isolated execution environments before they are admitted to production. The use of SafeTensors

format over PyTorch's default pickle-based serialization significantly reduces the attack surface by eliminating arbitrary code execution during deserialization, and organizations should require SafeTensors format for all newly adopted models where technically feasible.

6.3 Repository and Distribution Controls

Organizations should establish an internal model registry that serves as a controlled distribution point for AI models used in development and production. Rather than allowing development systems and CI pipelines to fetch models directly from Hugging Face or other public repositories, all model consumption should be routed through an internal registry that performs verification, scanning, and approval before models are made available internally. This mirrors the internal package proxy pattern that security-mature organizations apply to npm, PyPI, and other package registries — an established pattern from traditional software supply chain security that applies directly to AI model distribution.

For models sourced through cloud AI platform catalogs, organizations should not assume that the cloud provider has fully validated the models it offers against all current threat vectors. The namespace reuse demonstration against Google Vertex AI and Microsoft Azure AI Foundry established that cloud model catalogs can contain orphaned model paths that are vulnerable to re-registration attacks [8][16]. Organizations should apply their own verification controls to models sourced from cloud catalogs and should engage their cloud AI platform vendors to understand what verification is applied to catalog entries and what update processes exist when catalog entries are found to contain security issues.

6.4 Agentic System Controls

AI agent deployments require specific controls that address the autonomous tool selection and invocation behaviors that create supply chain exposure in agentic contexts. Agents that can dynamically select and invoke skills from public registries should operate in sandboxed execution environments that restrict filesystem access, network reach, and credential visibility to the minimum required for each specific workflow. This least-privilege principle is well-established in software security and applies with equal force to agentic AI deployments.

Organizations should maintain an allowlist of approved skills and tools for each agent deployment, preventing agents from autonomously invoking skills that have not been reviewed and approved. Dynamic skill discovery from public registries at runtime should be disabled in production environments unless the specific operational requirement cannot be met otherwise; where dynamic discovery is genuinely required, it should be subject to runtime verification against a cryptographic signature or hash committed at configuration time. Agentic deployments should be monitored for unexpected tool invocations, unusual network egress patterns, and credential access events that would indicate a malicious skill execution.

6.5 Governance and Organizational Practices

Technical controls are necessary but insufficient. Effective AI supply chain security requires organizational practices that maintain oversight of AI dependency decisions, ensure that security review is applied to new model adoptions, and establish clear processes for responding to disclosed vulnerabilities in AI model dependencies. Organizations should designate ownership for AI supply chain risk — whether in a dedicated AI security function, as an extension of existing application security responsibilities, or within the risk management function — and ensure that that owner has both the authority to require verification practices and the information needed to identify when new disclosures affect the organization's model inventory.

Incident response plans should be extended to address AI supply chain events. These include scenarios in which a model the organization depends on is found to contain malicious content, in which a model namespace the organization references is found to have been re-registered, and in which a dependency library used by a model is found to have a remotely exploitable vulnerability. For each scenario, the organization should have pre-established procedures for identifying affected systems, isolating the exposure, deploying a remediated or alternative model, and communicating with affected stakeholders. The speed with which namespace reuse attacks can compromise cloud AI platforms — potentially in hours — requires that response procedures be executable faster than traditional patch management timelines permit.

7. CSA Framework Alignment

The controls and practices described in Section 6 align to existing CSA frameworks that provide both the conceptual vocabulary and the audit mechanisms for implementing and assessing AI supply chain risk management. Organizations that have already adopted these frameworks can integrate AI model repository risk management as an extension of their existing control programs.

The AI Controls Matrix (AICM) v1.0, the foundational CSA framework for AI security governance, addresses supply chain risk through its supply chain security domain, which covers model provenance verification, dependency management, and third-party model vetting [22]. The AICM's model provider controls are directly applicable to the verification and scanning practices described in Section 6.2, and its AI customer controls address the organizational responsibilities for managing model dependencies and maintaining AI system inventories. Organizations using the AICM should assess their current coverage of supply chain controls against the specific threat vectors documented in this paper — malicious uploads, namespace reuse, shared dependency vulnerabilities, and agentic skill registry attacks — and use gaps identified in that assessment to prioritize control investments.

MAESTRO, CSA's threat modeling framework for agentic AI systems, provides the analytical structure for reasoning about the agentic supply chain attack surface described in Section 2.4 and Section 4.3. MAESTRO's threat categories address both the adversarial compromise of AI components and the autonomous behaviors of AI agents that amplify the impact of compromised components. Organizations deploying agentic AI systems should conduct MAESTRO-based threat models that specifically include the skill and tool registry dependencies of their agent deployments, treating each registry and each dynamically invocable skill as a potential supply chain attack vector.

The Cloud Controls Matrix (CCM) provides additional coverage through its supply chain management and application and interface security domains. CCM controls addressing software supply chain integrity, vulnerability management, and third-party risk management apply to AI model dependencies with appropriate adaptation: the "software" in software supply chain should be interpreted to include AI model artifacts, framework libraries, and agentic skills, and vulnerability management processes should incorporate the AI-specific vulnerability sources — model scanning reports, library CVE databases, platform security bulletins — that are relevant to AI model supply chain risk.

CSA's Zero Trust guidance reinforces the principle of verified trust rather than assumed trust that is central to addressing namespace reuse and shared dependency attacks: trust in a model identifier is not a substitute for cryptographic verification of the model artifact at that identifier, and trust in a cloud AI platform is not a

substitute for organizational verification of the specific models that platform serves. Applying Zero Trust principles to AI model consumption means treating every external model source as untrusted until verified, regardless of the reputation of the platform through which it is accessed.

8. Conclusions

The AI model repository ecosystem has achieved a concentration of dependency that represents a structural security challenge distinct from, and in important ways more difficult than, prior software supply chain incidents. The breadth and speed with which the ecosystem consolidated around Hugging Face as a primary distribution channel, PyTorch as a dominant model format, and shared libraries like Hydra as common infrastructure means that adversaries have a leverage ratio — attacker effort to victim reach — comparable to, and in some respects exceeding, that seen in prior software supply chain incidents such as SolarWinds and Log4Shell.

The documented incidents from 2024 through early 2026 demonstrate that adversaries have recognized this leverage and are actively exploiting it. Malicious model uploads with pickle-based payloads, scanner evasion through compression format variation, namespace re-registration attacks affecting both direct Hugging Face consumers and downstream cloud AI platforms, shared dependency vulnerabilities creating ecosystem-wide exposure, and coordinated skill registry poisoning operations are not theoretical risks — they are documented, executed attacks with concrete victim populations.

The transition toward agentic AI deployment extends this attack surface into a new dimension where autonomous tool selection and machine-speed code execution eliminate the human review steps that might otherwise provide a last-resort detection opportunity. As agentic systems gain broader deployment, the importance of verified trust in AI model and skill supply chains will only increase.

The path forward requires action across three levels. At the technical level, organizations must implement cryptographic verification, version pinning, artifact scanning, internal model registries, and agentic isolation controls. At the governance level, they must build AI model dependencies into existing risk management programs, extend incident response procedures to address supply chain events, and establish ownership of AI supply chain risk within the organization. At the ecosystem level, the standards community, platform operators, cloud AI providers, and regulatory bodies must accelerate the development and adoption of model signing infrastructure, permanent namespace reservation, scanner-resistant verification approaches, and audit requirements proportionate to the systemic importance of AI model repositories.

CSA's AI Controls Matrix, MAESTRO, and Cloud Controls Matrix provide the frameworks within which organizations can structure these efforts. The threat described in this paper is neither speculative nor distant — it is active, documented, and growing. Organizations that treat AI model supply chain security as equivalent in importance to traditional software supply chain security will be better positioned to absorb the incidents that will continue to occur as the ecosystem matures.

References

[1] Hugging Face. "State of Open Source on Hugging Face: Spring 2026." Hugging Face Blog, Spring 2026.

[2] WorldMetrics. "Hugging Face Statistics: Market Data Report 2026." WorldMetrics, 2026.

[3] JFrog Security Research. "Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor." JFrog Blog, February 2024.

[4] Open Source For You. "Hydra Dependency Creates Systemic AI Security Risk Across Hugging Face Models." Open Source For You, January 2026.

[5] The Hacker News. "Over 100 Malicious AI/ML Models Found on Hugging Face Platform." The Hacker News, March 2024.

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

[7] ReversingLabs. "Malicious ML Models Discovered on Hugging Face Platform." ReversingLabs Blog, February 2025.

[8] Palo Alto Networks Unit 42. "Model Namespace Reuse: An AI Supply-Chain Attack Exploiting Model Name Trust." Unit 42 Blog, Palo Alto Networks, September 3, 2025.

[9] Repello AI. "ClawHavoc: Inside the Supply Chain Attack That Targeted 300,000 AI Agent Users." Repello AI Blog, February 2026.

[10] Snyk. "Snyk Finds Prompt Injection in 36%, 1467 Malicious Payloads in a ToxicSkills Study of Agent Skills Supply Chain Compromise." Snyk Blog, February 2026.

[11] U.S. Department of Defense. "Artificial Intelligence and Machine Learning Supply Chain Risks and Mitigations." DoD, March 2026.

[12] Crowell & Moring. "CMMC for AI? Defense Policy Law Imposes AI Security Framework and Requirements on Contractors." Crowell & Moring Client Alert, January 2026.

[13] Financial Stability Board. "Monitoring Adoption of Artificial Intelligence and Related Vulnerabilities in the Financial Sector." FSB Report, October 2025.

[14] BleepingComputer. "Malicious AI Models on Hugging Face Backdoor Users' Machines." BleepingComputer, February 2024.

[15] Rescana. "Supply Chain Attack: Fake OpenAI Repository on Hugging Face Distributes Infostealer Malware." Rescana, May 2026.

[16] SecurityWeek. "AI Supply Chain Attack Method Demonstrated Against Google, Microsoft Products." SecurityWeek, September 2025.

[17] Palo Alto Networks Unit 42. "Remote Code Execution With Modern AI/ML Formats and Libraries." Unit 42 Blog, Palo Alto Networks, January 13, 2026.

[18] Open Source Security Foundation. "An Introduction to the OpenSSF Model Signing (OMS) Specification." OpenSSF Blog, June 2025.

[19] Sigstore / GitHub. "sigstore/model-transparency: Supply Chain Security for ML." GitHub, 2025.

[20] Sonatype. "2026 State of the Software Supply Chain Report." Sonatype, 2026.

[21] InformationWeek. "Your AI Vendor Is Now a Single Point of Failure." InformationWeek, 2025.

[22] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.0 Introduction Guidance." Cloud Security Alliance, 2025.