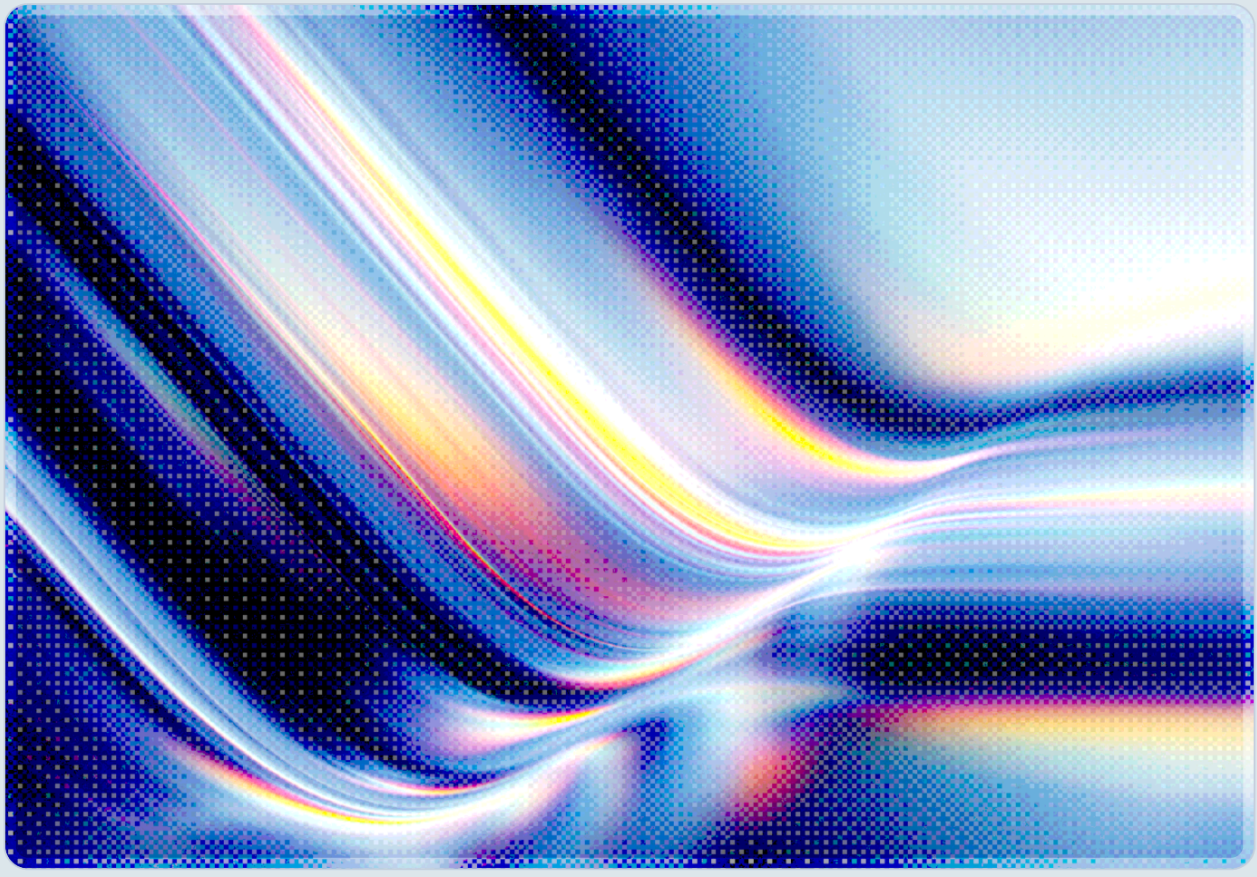
CSAI Foundation | Cloud Security Alliance

Model Hub Typosquatting: Malicious AI in Enterprise Pipelines

Security Risks from Hugging Face Namespace Abuse and Malicious Model Uploads

2026-05-12

AI-assisted Rapid Research



CSAI | cloud security alliance®

Key Takeaways

The Hugging Face model hub, which hosts over one million openly accessible machine learning models [9], has become an active target for supply chain attackers deploying techniques borrowed from the software package ecosystem. Academic researchers analyzing the platform found 1,574 potentially malicious typosquatting models, and independent security teams have documented real-world campaigns delivering reverse shells and credential-stealing malware through files that superficially resemble legitimate model releases [1][2][3]. The primary attack surface is Python's pickle serialization format, which permits arbitrary code execution at model load time and remains the dominant storage format for PyTorch models despite the existence of safer alternatives [4][5]. Enterprise security teams face a compounding risk: traditional software composition analysis tools were not designed to inspect model weight files, and threat actors have already demonstrated techniques that bypass platform-level scanning controls [6][7]. Immediate risk reduction requires prohibiting the use of pickle-format models from unvetted sources, adopting cryptographic verification for approved model artifacts, and integrating model provenance checks into ML pipelines as a required gate rather than an optional review.

Background

The Hugging Face Hub launched as a community repository for sharing natural language processing models and has grown into the dominant platform for open-source AI model distribution, hosting models from individual researchers, academic institutions, and major technology companies. Its design mirrors package managers such as PyPI and npm in that any registered user can publish artifacts under a namespace that closely resembles an existing, legitimate identity. This openness—which enables rapid scientific collaboration—also creates the same namespace collision risks that have affected open-source software repositories for a decade. Many organizations, lacking model-specific vetting processes, treat high download counts or familiar namespace names as informal authenticity signals when sourcing pre-trained models from Hugging Face—a behavior that threat actors have explicitly exploited [2][4].

What distinguishes model repositories from package managers is the file format carrying the attack payload. Rather than malicious code embedded in a setup script or build file, adversaries embed executable payloads inside serialized model weight files. The Python pickle format, used by PyTorch's default `.pt` and `.bin` formats, was designed for general Python object serialization and permits arbitrary class instantiation during deserialization. A model file that visually appears as a collection of

numeric tensors can silently execute a reverse shell, exfiltrate credentials, or implant backdoors into the model's inference behavior the moment a data scientist or automated pipeline calls `torch.load()`. For repositories distributing pickle-format model files, even a careful manual review of a repository's README and metadata provides no assurance that loading the model is safe.

Enterprise AI adoption has expanded the potential blast radius of these attacks, as production pipelines now operate with broader network access and data privileges than the research environments where this risk was first identified. As organizations move from pilot experiments to production ML systems, automated pipelines pull dependencies—including base models—with the same logic used to install software libraries. If a model is pulled from Hugging Face without integrity verification and executed in a context with access to internal networks, secrets managers, or inference infrastructure, a malicious payload gains a foothold that may be invisible to conventional endpoint detection tools.

Security Analysis

The Typosquatting Threat Surface

The first systematic empirical measurement of typosquatting in the Hugging Face ecosystem was published at the Internetware 2025 conference. Researchers analyzed 1,020,755 models against the 100 most-downloaded legitimate ones, 219,812 datasets against the 100 most-trending ones, and 127,011 organization namespaces [1]. Across these three categories, the study identified 1,574 potentially malicious squatting models—of which 10.4% exhibited confirmed suspicious or harmful characteristics—along with 625 typosquatted datasets where 42.2% showed evidence of intentional impersonation, and 302 organizations with squatting patterns in their namespace registrations, with 4.8% demonstrating explicit malicious intent [1]. If this ecosystem follows the trajectory observed in npm and PyPI, the current scale may foreshadow larger supply chain incidents—though the structural differences between model hubs and package registries mean the analogy should not be taken as deterministic.

Typosquatting techniques in this context include character substitution (replacing "1" with "l"), homoglyph attacks using Unicode characters that render identically to ASCII, hyphen insertion, and plural or version-number suffixes appended to legitimate model names. Attackers also register organization namespaces that closely resemble well-known AI labs, creating a convincing top-level brand impression even when individual model names differ. A developer copying a model identifier from documentation or a community forum post is unlikely to scrutinize every character of a namespace like `anthfu` versus `anthropic`, particularly when both resolve to functional repositories on the same platform.

Documented Attack Campaigns

The threat has moved well beyond proof of concept. In a high-profile campaign analyzed by multiple security vendors, a repository named `Open-OSS/privacy-filter` impersonated an official OpenAI release and accumulated 244,000 downloads before being identified and removed [2]. The malicious repository included a `loader.py` file that fetched and executed credential-stealing malware targeting Windows hosts, achieving lateral movement potential and cryptocurrency wallet access in affected developer environments. The attack relied entirely on naming proximity and positioning within trending lists; no technical exploit of the Hugging Face platform itself was required [2].

Separately, JFrog security researchers scanning PyTorch and TensorFlow Keras models on Hugging Face identified approximately 100 repositories containing malicious payloads, of which PyTorch models represented the large majority [3]. One notable example was `baller423/goober2`, a repository containing a model file that used Python's `__reduce__` pickle method to establish a reverse shell connection to a hardcoded external IP address on load [3][8]. The JFrog team subsequently built continuous monitoring infrastructure to scan new model uploads multiple times daily, and disclosed their findings directly to Hugging Face. This scanning partnership expanded into a broader collaboration: as of the most recent public update, Protect AI—working alongside Hugging Face—had scanned more than four million models and identified approximately 352,000 unsafe or suspicious issues distributed across 51,700 models [9].

The Pickle Attack Surface and Detection Evasion

The technical mechanism underlying most of these attacks is Python's pickle serialization format. A pickle file is not a static data container; it is a sequence of opcodes interpreted by the pickle virtual machine, which can reconstruct arbitrary Python objects by calling any importable class or function. An attacker constructs a pickle payload that calls `os.system()`, `subprocess.Popen()`, or any other executable routine, and embeds it within what otherwise appears to be a legitimate model file. Repositories containing only pickle-format models are downloaded more than 400 million times per month globally, indicating the scale at which this attack surface is exercised [16].

The ecosystem's primary platform-level control, PickleScan, has proven bypassable. As reported by The Hacker News, ReversingLabs documented the "NullIfAI" technique in early 2025: by compressing model files using 7z rather than PyTorch's default ZIP format, attackers produced files that PickleScan could not process, while PyTorch's loader could still partially deserialize them—executing the payload embedded at the start of the pickle byte stream before the loader encountered and reported an error [6]. Hugging Face patched PickleScan within 24 hours of disclosure, but this incident demonstrated the adversarial dynamic now in place: platform defenses are being specifically researched and circumvented. Sonatype

subsequently identified four additional bypass techniques against PickleScan, including bit-flipping specific ZIP header flags to conceal malicious pickle content inside PyTorch archives [7]. CVE-2025-46417 documents another variant that allows malicious models to bypass scanners and exfiltrate sensitive data [10].

Beyond outright malware delivery, researchers at Trail of Bits described "Sleepy Pickle" as a proof-of-concept attack that modifies a model's weights in-place during deserialization rather than executing an obvious payload [5]. Because the technique alters only a small subset of weight values, it adds less than 0.1% overhead to the pickle file and is imperceptible to static inspection [5]. The resulting model behaves normally on standard benchmarks while producing attacker-controlled outputs for targeted inputs, enabling backdoor insertion into production inference systems without triggering integrity alarms. While documented as a proof-of-concept rather than a confirmed observed campaign, the technique illustrates a class of attack that existing detection tooling is structurally unable to catch.

Namespace Hijacking After Account Deletion

Researchers at Palo Alto Networks Unit 42 identified a structural vulnerability in the Hugging Face identity system: when an author deletes their account, their namespace becomes immediately available for re-registration. An adversary who registers a deleted author's username and uploads poisoned versions of that author's formerly popular models gains automatic traffic from any pipeline still referencing the original model path—because the path resolves successfully and the repository appears populated [11]. This attack requires no typosquatting at all; the original model identifier is reused verbatim. Organizations using hardcoded model identifiers in pipeline scripts are exposed to this vector, since those scripts will continue fetching from the same path after the original author has left the platform and the namespace has been re-registered.

Detection Gaps in Enterprise Tooling

Traditional software composition analysis platforms were built to inspect dependency manifests, SBOM entries, and container image layers—not the behavioral properties of binary neural network artifacts. Applying SCA tooling to Hugging Face model pulls catches known-bad hashes when signatures have been published, but provides no assurance against novel payloads in new model uploads. Approximately 15% of repositories containing only pickle-format models include at least one model that cannot be processed by PyTorch's `weights_only=True` unpickler, the recommended safe-loading mode—indicating that safe loading mode itself cannot be universally applied without breaking existing model loading workflows [16]. These factors create conditions in which compromises may go undetected in environments without dedicated model scanning—even though the incidents documented in this note were ultimately identified

through external security research rather than organizational detection capability. The asymmetry between attacker research investment and enterprise detection readiness suggests that undetected incidents are plausible in organizations that have not implemented proactive model scanning.

Recommendations

Immediate Actions

Enterprise ML teams should treat any pickle-format model obtained from an external repository as untrusted code until proven otherwise. Automated pipelines must be audited to identify every location where models are fetched from Hugging Face or similar repositories, and those fetch operations should be suspended or gated behind manual review for any model not already in an internal approved registry. Existing model inventories should be spot-checked: model files loaded with `torch.load()` without `weights_only=True` should be identified and, where possible, reloaded in safe mode or converted to a safer format such as Safetensors or ONNX. Any pipeline environment with access to internal secrets, production databases, or corporate networks should be treated as a high-value target requiring isolation from internet-accessible model downloads.

Teams should also audit Hugging Face model identifiers hardcoded in pipeline scripts and configuration files. Any identifier referencing an account namespace that is no longer active on the platform represents a namespace hijacking risk and should be resolved against an internal, pinned copy of the artifact rather than a live platform pull.

Short-Term Mitigations

Establishing an internal model registry with explicit allow-listing provides strong structural protection by centralizing vetting and reducing the attack surface from live platform pulls. Rather than permitting pipelines to pull models directly from external sources, organizations should require that all models pass through an internal registry after inspection—analogous to the practice of proxying package registries for software dependencies. Model intake into this registry should include: hash verification of the artifact against a published checksum (where one exists), scanning with multiple tools including ModelScan [12], conversion from pickle format to Safetensors where technically feasible, and a record of the source URL, download date, and reviewing engineer. For models that cannot be converted due to custom architecture requirements, a documented exception with compensating controls (network isolation, sandboxed execution) should be required.

Pipeline code should enforce the `weights_only=True` parameter in all `torch.load()` invocations as a baseline policy. Security teams should deploy behavioral monitoring on model inference services to detect anomalous outbound connections, unexpected subprocess spawning, or unusual file system writes that may indicate payload execution from a previously loaded model. Model file hashes should be captured in pipeline logs at load time to support forensic reconstruction if an incident is later discovered.

Strategic Considerations

Longer-term supply chain integrity for AI artifacts requires extending software bill of materials practices to model provenance. An AI BOM should capture the model's source repository and commit hash, the format and serialization method, the checksum of the weights file, the identity of the internal reviewer who approved intake, and the downstream components that consume the model. This aligns directly with the Supply Chain Management, Transparency, and Accountability (STA) domain controls defined in the CSA AI Controls Matrix. Organizations pursuing formal AI governance programs should treat model provenance documentation with the same rigor applied to library dependencies.

Procurement and vendor management teams evaluating AI platform providers should assess what scanning and integrity controls the provider applies to third-party model artifacts, whether namespace reclamation policies prevent squatting on deleted accounts, and what cryptographic signing infrastructure is available for model authenticity verification. Hugging Face has introduced optional model signing capabilities and maintains the Protect AI scanning partnership, but these controls are not uniformly applied to all models and cannot substitute for organizational intake controls on the consuming side.

CSA Resource Alignment

This research note addresses threats that span multiple CSA framework domains. The CSA AI Controls Matrix (AICM) provides the most directly applicable guidance through two domains: the Model Security (MDS) domain, which specifies controls for model artifact scanning, integrity verification, model signing, and continuous monitoring of deployed models; and the Supply Chain Management, Transparency, and Accountability (STA) domain, which defines controls for AI service inventory, third-party component vetting, and Bill of Materials requirements for AI services. Enterprises implementing the AICM's MDS-series controls are well-positioned to enforce the intake procedures and format restrictions recommended in this note [13].

The CSA MAESTRO framework for agentic AI threat modeling addresses supply chain risk at its Layer 1 (Foundation Models), which defines the base risk surface introduced when an organization consumes a third-party pre-trained model. MAESTRO's analysis makes explicit that threats introduced at the foundation model layer can cascade across all subsequent layers of an agentic architecture—into data operations, agent frameworks, and deployment infrastructure—making model provenance a foundational security concern for agentic architectures [14]. The namespace hijacking and Sleepy Pickle scenarios described in this note represent exactly the kind of cross-layer threat chains that MAESTRO was designed to surface during architecture review.

The CSA AI Model Risk Management Framework, which structures model governance around Model Cards, Risk Cards, and Scenario Planning, provides an organizational process for documenting the risk profile of each model in use, including its sourcing provenance, format vulnerabilities, and compensating controls—supporting audit and incident response workflows when a compromise is suspected [15]. Organizations registered in the CSA STAR program can use the AICM control attestations to communicate their supply chain security posture to customers and auditors, distinguishing organizations that have implemented structured model intake processes from those relying solely on platform-level controls.

References

[1] Lin, Y. et al. "Exploring Typo Squatting Threats in the Hugging Face Ecosystem." Proceedings of the 16th International Conference on Internetware (Internetware 2025), ACM, 2025.

[2] Swain, G. "Malicious Hugging Face model masquerading as OpenAI release hits 244K downloads." CSO Online, 2025.

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

[4] Boi, A. et al. "Paws in the Pickle Jar: Risk & Vulnerability in the Model-sharing Ecosystem." Splunk Security Blog, 2024.

[5] Trail of Bits. "Exploiting ML models with pickle file attacks: Part 1." Trail of Bits Blog, June 2024.

[6] Lakshmanan, R. "Malicious ML Models on Hugging Face Leverage Broken Pickle Format to Evade Detection." The Hacker News, February 2025.

[7] Sonatype Security Research. "Exposing 4 Critical Vulnerabilities in Python Picklescan." Sonatype Blog, 2024.

[8] Toulas, B. "Malicious AI models on Hugging Face backdoor users' machines." BleepingComputer, February 2024.

[9] Hugging Face. "4M Models Scanned: Protect AI + Hugging Face 6 Months In." Hugging Face Blog, 2024.

[10] NIST National Vulnerability Database. "CVE-2025-46417." NIST NVD, 2025.

[11] Saraf, I. and Balassiano, O. "Model Namespace Reuse: An AI Supply-Chain Attack Exploiting Model Name Trust." Palo Alto Networks Unit 42 Blog, September 2025.

[12] Protect AI. "ModelScan - Protection Against Model Serialization Attacks." GitHub, 2023.

[13] Cloud Security Alliance. "AI Controls Matrix." CSA, July 2025.

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

[15] Cloud Security Alliance. "AI Model Risk Management Framework." CSA, 2024.

[16] Brown University and Columbia University researchers. "PickleBall: Measuring the Pickle Deserialization Attack Surface in Machine Learning Ecosystems." arXiv preprint arXiv:2508.15987, 2025.