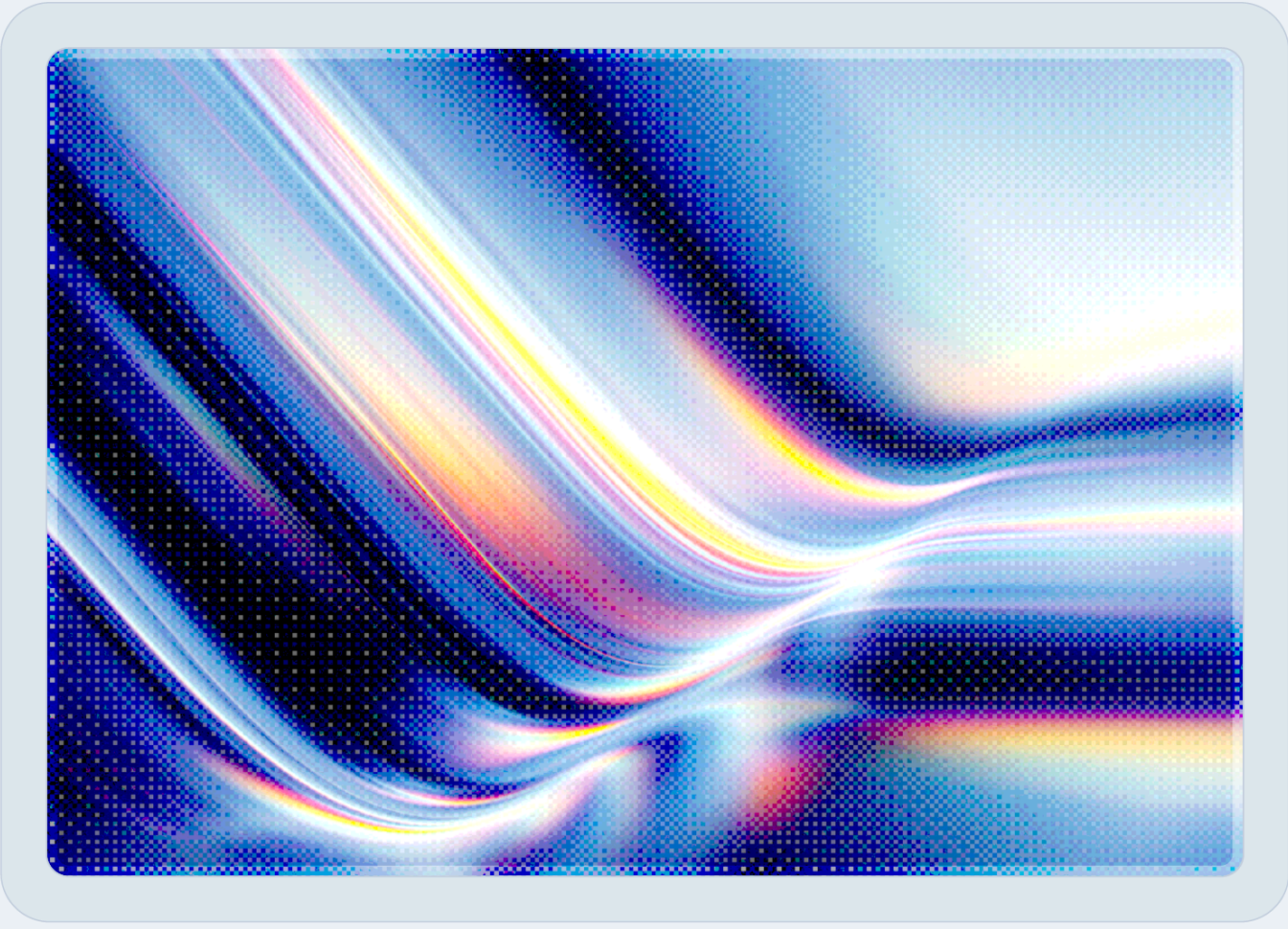
CSAI Foundation | Cloud Security Alliance

# Takedowns That Don't Hold: Model Hub Re-Uploads and Agent Lures

## Re-Upload Persistence on Model Hubs and Lures Built for AI Coding Agents

2026-10-31

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

**© 2026 Cloud Security Alliance. Some rights reserved.**

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

*This document was generated with AI assistance and has not undergone official CSA review and approval processes.*

## Key Takeaways

A cryptomining and backdoor campaign on Hugging Face came back under new accounts within days of its first takedown. Pluto Security documented 31 malicious repositories across seven Hugging Face accounts, published in two waves. The 19 second-wave repositories were still live eight days after the researchers reported them, and downloads across the campaign and its linked models exceeded 32,000 as of October 28, 2026 [1]. After Hugging Face disabled the flagship repository, BleepingComputer confirmed that a near-identical copy uploaded by a newer account on October 25 was still serving the same miner and reverse shell [2]. Organizations should therefore treat a takedown notice as the start of their own response, not as proof that the threat is gone.

The campaign also shows how far behavior-blind scanning falls short. Its payload was ordinary Python that the Transformers library executes when a model is loaded with `trust_remote_code=True`; it used no poisoned weights and no pickle exploit. One flagship repository contained a 42-byte weights file with no tensors at all [1]. Platform scanners look for known malware signatures and unsafe serialization formats, and the second-wave flagship scanned clean while it accumulated downloads [2].

Three recent reports suggest a shift toward lures written for AI coding agents rather than for people. A fake Solana starter-kit repository created in November 2025, whose only file is a README containing a PowerShell install command, reportedly ranks first when AI coding agents search for that kind of template [2][9]. Separately, research published in late October 2026 found that a planted `AGENTS.md` file in a Downloads folder got OpenAI Codex to execute a test payload in 90% of laboratory trials [7]. In July 2026, Island reported more than 800 malicious GitHub repositories posing as AI Skills or MCP servers, and in its tests mainstream AI assistants surfaced malicious campaign repositories without being given a link [10].

Both threads share a root cause: loading or reading an artifact is treated as consent to execute it. Defenders should move trust decisions away from where an artifact is hosted and toward pinned, reviewed content. Agents and model loaders should run in environments where executing hostile code has a small blast radius.

# Background

Model hubs have hosted malicious artifacts for years, and the techniques have changed as each round of defenses arrived. In 2024, JFrog reported roughly 100 malicious PyTorch and TensorFlow Keras models on Hugging Face. Some used pickle's `__reduce__` mechanism to open a shell on the machine that loaded them [5]. In February 2025, ReversingLabs described "nullifAI," which used deliberately broken pickle files to slip past the Picklescan tool. Picklescan, the open-source scanner Hugging Face uses, was subsequently updated to handle malformed files [6]. Each episode pushed platforms toward format-level controls, such as the Safetensors format and pickle scanning, and toward the antivirus scan that Hugging Face runs on every committed file smaller than 2 GB [3].

The `trust_remote_code` path avoids these controls. Transformers requires custom model architectures to ship their own Python modeling code, and users must pass `trust_remote_code=True` to load them [4]. The `auto_map` field in a repository's `config.json` names the Python file to import, and any top-level code in that file runs before the model is built. In the Pluto campaign, the model files were incidental. The repositories posed as small Qwen2 test checkpoints, and the payload lived entirely in the imported module [1][2]. A scanner that checks weights for unsafe serialization finds nothing wrong, because nothing is wrong with the weights.

The second background development concerns who reads repository content. AI coding agents now search the web and package registries, read READMEs, and follow project-level instruction files such as `AGENTS.md` and `CLAUDE.md` with the user's own permissions. Research on package hallucination had already shown that code-generating models regularly produce plausible but nonexistent dependency names. Spracklen et al. recorded 205,474 unique hallucinated package names, with average hallucination rates of at least 5.2% for commercial models and 21.7% for open-source models [11]. Attackers have since moved past waiting for agents to hallucinate. They now publish content built to be found by agent search and to read like legitimate instructions.

# Security Analysis

## Anatomy of the Load/Mine Re-Upload Campaign

Pluto Security's report, published October 29, 2026, traces the campaign from an operator account created in April 2026 to a first wave of repositories that began September 21–22 and continued through October 12 [1]. The first-wave code installed a fileless miner for the Pearl (PRL) cryptocurrency. It disguised the miner as a kernel worker process, kept it alive with a watchdog and shell startup hooks, killed

competing miners, and added an attacker SSH key to `authorized_keys`. Pluto linked a test account to the operator through leftover backup files and reported the accounts to Hugging Face and to China's ModelScope, and both platforms removed them [1][2].

The second wave arrived within days under four new accounts, with a fifth (`htt0118`) added on October 25 [1]. It used revised loaders and replaced the hardcoded SSH key with reconnecting reverse shells, a change Pluto interprets as evidence that the operator adapted to the first takedown [1]. According to Pluto, the operator published a container-escape script about 8.5 hours after Pluto sent its disclosure email to the platforms, and publishing continued after the disclosure [1]. Downloads of the second-wave flagship rose from 7,354 when Pluto disclosed it to Hugging Face, around October 20, to 28,412 by October 28 [1][2]. While the researchers watched, nine machines that appeared to belong to a shared research GPU cluster began mining for the operator. They stopped about six minutes later, which is consistent with an administrator killing the jobs [2].

The table below summarizes the two waves as Pluto reported them.

| Attribute | Wave 1 (removed) | Wave 2 (live at time of report) |
|---|---|---|
| Repositories | 12 across two Hugging Face accounts and one ModelScope account | 19 across five Hugging Face accounts |
| Remote access | Hardcoded ed25519 SSH key | Reconnecting reverse shells on ports 443 and 9443 |
| Downloads | 728 | 29,090 |
| Additional tooling | Fileless miner, watchdog, rival-miner killer | Revised loaders, container-escape script |

*Source: Pluto Security [1]. Download counts measure downloads, not confirmed infections.*

These figures matter more for the pattern than for their size. Removing the first wave did not end the campaign [1]. Second-wave downloads were roughly forty times those of the first, although a single repository accounts for 97.7% of them and much of that growth followed public disclosure, when researchers, journalists, and scanners were likely fetching it, so the figure is a weak proxy for infections [1]. The flagship's replacement copy was live on the day BleepingComputer published [2]. The operator's mining wallet also stayed active after the repositories were removed, because takedowns on a hub do nothing to the attacker's infrastructure elsewhere [2]. Pluto attributes the activity to a financially

motivated, Chinese-speaking operator, citing code comments, command-and-control hosting, and activity times. This attribution is the researchers' assessment and has not been independently confirmed [2].

The campaign also targeted a particular kind of environment. BleepingComputer reports that the payload went after GPU hosts running self-hosted inference servers such as Xinference and GPUStack. It reported its status through the Xinference model registry [1][2]. Pluto's target list also includes rented GPU machines, shared notebooks, and CI jobs [1]. In CSA's assessment, hosts like these frequently have accelerators, broad network reach, and credentials for model registries and cloud storage, and they may be run by research teams outside central security monitoring. CSA's recent analysis of a critical unauthenticated remote code execution flaw in Xinference's underlying RPC framework reached a similar conclusion about self-hosted inference as an initial-access target [12].

## Why Platform Takedowns Do Not Hold

Platform takedowns act on individual artifacts and accounts, while the attacker operates as a continuing campaign. Creating a new account on a public hub costs little, and copying a repository costs even less. A detection that depends on a researcher noticing and reporting each repository will therefore lag behind an operator who can republish in minutes. Pluto's observation that 19 reported repositories stayed live for eight days [1] suggests that review capacity, not detection, was the limiting factor in this case. That inference rests on one campaign and should be tested against others.

The scanning problem makes the lag worse. Hugging Face documents antivirus scanning of committed files [3], and the platform also uses partner scanners from Protect AI and JFrog [2]. Pluto reports that none of these flagged the malicious Python files. One partner tool classified the payload file as "not a machine-learning model" [1][2]. That result is accurate, but it does not help anyone decide whether the file is safe. Pluto observes that the partner scanners focus on weight formats [1], and model scanning has largely been built to answer whether serialized weights can execute code. A `trust_remote_code` repository answers that question up front, because it ships Python that executes by design. To decide whether that code is safe, a scanner has to analyze what it does: whether it spawns subprocesses, opens sockets, calls `ctypes`, or writes persistence hooks. Pluto recommends exactly this kind of static check before `trust_remote_code` is enabled [1].

For downstream organizations, a takedown leaves two gaps. First, any host that loaded the repository with `trust_remote_code` enabled before removal may remain compromised and should be treated as such, because stopping the miner does not remove the SSH keys, reverse shells, or startup hooks [2].

Second, a pipeline that resolves models by repository name rather than by pinned commit can pick up a near-identical copy under a new namespace if anyone updates a reference or follows a search result. Takedowns reduce new exposure on the original repository and do little about either gap.

## Lures Written for Agents

Agent lures move the same logic from model hubs to code repositories and instruction files. The fake starter kit documented by Air Security, `SolanaUpdateTools/solana-starter-kit`, which Air links to a campaign it calls SolanaCry, was created in November 2025 and had 286 stars when examined. It advertises a full dApp template but contains only a README [2][9]. The README's single installation step is a PowerShell one-liner that assembles a URL from fragments and runs a script in memory. The script installs a covertly configured NetSupport Manager client that connects to attacker infrastructure at every login [2][9]. Air reports that the repository ranks as the top result when AI coding agents search for a Solana starter kit [2]. Air found the payload URL returning 404 when it checked. BleepingComputer recovered the archived payload from the Wayback Machine, confirmed Air's analysis, and observed the repository still live at the time of writing [2]. The repository therefore remains a lure that the attacker can re-arm by repointing the payload host.

The `AGENTS.md` technique reported by Forever Security reaches the agent's context through the filesystem instead of through search. Researchers found a malicious `AGENTS.md` in a customer's Downloads folder that caused a coding agent to try installing the npm package `function-color` [7]. Agents load instruction files from their working directory automatically. Forever's telemetry showed more than 1,300 Claude Code sessions launched from Downloads, and Chrome, Edge, and Firefox save downloads there by default without prompting [7]. In laboratory tests, a harmless payload succeeded against Codex in 90% of trials. A separate `CLAUDE.md` variant redirected Claude Code's API traffic to a researcher-controlled server and exposed its OAuth token [7]. Coverage of the Claude Code results differs: BleepingComputer reports roughly 50% success for code-executing tool calls [7], while other summaries separate a roughly 50% endpoint-redirection rate from a lower same-session execution rate [8]. Readers should consult the primary research before relying on a specific figure. The researchers did not determine how the file reached the customer's machine [7], and they argue that OpenAI and Anthropic have nothing to patch because the agents behaved as designed [7]. That framing is the researchers' view rather than a settled conclusion.

Island's July 2026 "AgentBaiting" research shows the scale such lures can reach. Island tracked about 7,600 malicious GitHub repositories from about 6,600 profiles. More than 800 posed as AI Skills or MCP servers, and campaign listings appeared more than 600 times across public MCP and Skill registries [10]. The repositories delivered the SmartLoader loader and the StealC infostealer. Variants reportedly locate their command server through a value stored in a Polygon smart contract, which a repository takedown

cannot remove [10]. In Island's tests, all three mainstream assistants it examined surfaced malicious campaign repositories without being given a link [10]. Asked for a free Walmart MCP server, Gemini recommended a malicious repository first and ChatGPT called it the best starting point. In a separate test asking for a free prompt Skill, Claude Code recommended a benign repository but also presented a malicious one as an alternative and repeated its install steps in one run, while refusing it in others [10].

## Convergence: Content as Executable Instruction

The two threads share a structure, summarized below.

| Dimension | Model hub re-upload (Load/Mine) | Agent lures (SolanaCry, AGENTS.md, AgentBaiting) |
|---|---|---|
| Discovery channel | Hub search, plausible model names | Agent web search, registry listings, filesystem auto-load |
| Execution trigger | `trust_remote_code=True` imports repository Python | Agent follows README or instruction-file commands |
| What the trust decision relies on | Platform scanning of signatures and serialization formats | Repository reputation, stars, registry presence |
| Takedown resilience | New accounts and near-identical copies within days | Live repositories, repointable payload hosts, blockchain-based C2 lookup |
| Primary victim environment | GPU inference hosts and research clusters | Developer workstations with credentials and tokens |

In both cases, the decision to execute is made implicitly. A data scientist enables a flag so a model will load, and an agent treats repository text as operating instructions. Neither the hub's scanners nor the code host's reputation signals analyze the behavior of the code that will actually run. The two threads may also be converging. Agents increasingly fetch and load models as part of building applications, so an agent following a lure could just as easily be told to load a model with `trust_remote_code=True`. CSA has not observed this combination in the reporting reviewed for this note, so it remains an inference. Defenders should still assume it is plausible.

# Recommendations

## Immediate Actions

Organizations should search for the indicators Pluto published, including the `/tmp/.cch/` directory, unexpected `svc.py` processes, a `[kworker/u20:1]` process with mining arguments, and unfamiliar Xinference model registrations [1]. Any host that loaded the affected repositories with `trust_remote_code` enabled should be treated as compromised. Teams should remove shell hooks, cron entries, `authorized_keys` additions, and registry entries, rotate reachable credentials from a clean machine, and rebuild the host if its integrity cannot be confirmed [1][2]. Xinference and similar inference APIs should require authentication and be unreachable from untrusted networks [1].

For coding agents, teams should stop launching agents from Downloads or other folders where browsers can drop files. They should audit stray `AGENTS.md` and `CLAUDE.md` files, inspect agent configuration such as `~/.codex/config.toml` and `.claude/settings.json` for unfamiliar entries, and rotate agent tokens if a planted file is found [7][8]. Approval prompts and sandboxing should stay enabled [7]. Security teams should also block the published SolanaCry infrastructure and hunt for unexpected NetSupport Manager installations under user application-data directories [2][9].

## Short-Term Mitigations

Disable `trust_remote_code` by default in shared environments and require an exception process to enable it. Before granting an exception, review the Python files named in `auto_map`, and pin the approved repository to a specific commit hash rather than a mutable name or branch [1]. Mirror approved models into an internal registry so that production pipelines never resolve models from a public hub at runtime. Provided the mirror's intake process itself pins hashes and requires review, a copied repository under a new namespace then cannot enter the pipeline without passing the same review and pinning process. Run model loading for untrusted or newly evaluated artifacts in isolated containers that hold no production secrets and have restricted egress. Pluto's container-escape script, which targets privileged containers, is a reminder that loader containers must not run privileged and that the isolation boundary itself needs hardening [1].

For agents, maintain a curated catalog of approved Skills, MCP servers, and plugins. Verify publishers rather than relying on stars or registry listings, and reject any capability whose installation requires running a downloaded executable or a piped shell command [10]. Monitor agent-initiated downloads, clones,

package installs, and changes to agent configuration directories [10]. Where agent frameworks support it, restrict package installation to an internal registry proxy that serves only allowlisted packages, so that hallucinated or lure-supplied package names fail closed.

### Strategic Considerations

Supply chain programs should plan for takedowns that fail. The governing assumption should be that any public artifact can reappear under a new identity, so organizations need to anchor trust to content hashes, signatures, and internal review records rather than to namespaces. In CSA's assessment, behavior-aware scanning of executable repository content, operator-level clustering of related accounts, and similarity detection for near-identical re-uploads are more likely to break the re-upload cycle than faster single-repository removal. Enterprises should ask their platform providers whether these capabilities exist. Agent vendors and enterprise policy owners should also settle where instruction-file trust ends. A file that an agent auto-loads from its working directory can carry authority comparable to the user's prompt. Where agents offer workspace-trust controls, those controls should stay enabled, and the grant of that authority should be explicit.

## CSA Resource Alignment

The campaign's focus on self-hosted inference servers is consistent with CSA's research note *Xinference Actor-Layer RCE: Self-Hosted Inference as Initial Access* [12]. That note treated Xinference deployments as an initial-access target because of an unauthenticated remote code execution flaw. The Load/Mine campaign reaches the same hosts through the model-loading path instead and abuses the Xinference registry for status reporting [1]. Read together, the two analyses suggest that inference infrastructure needs the network isolation, authentication, and patch discipline that organizations already apply to production application servers.

The agent-lure findings build on CSA's note *AI Agent Prompt Injection: The New CI/CD Supply Chain Threat* [13], which found that attacker-controlled text reaching an agent with execution privileges is a supply chain problem, not only a model-safety problem. The `AGENTS.md` and SolanaCry cases extend that finding from CI/CD events to the developer workstation and to agent web search. CSA's note *Markdown Is the New Executable: Instruction-File Poisoning of AI Coding Agents* [17] examines the `AGENTS.md` and `CLAUDE.md` technique in greater depth, and *AI Coding Agents as Attack Surface: MCP, Poisoning, and Miasma* [18] places repository-configuration poisoning and MCP abuse in the broader coding-agent

attack surface. CSA's note *Miasma and IronWorm: Self-Replicating Worms Targeting AI Credentials* [14] shows what typically follows a successful lure: harvesting AI coding tool credentials and spreading through package ecosystems.

For control mapping, the *AI Controls Matrix (AICM) v1.1* [15] provides the relevant supply chain, threat and vulnerability management, and application security control domains. These cover provenance verification for third-party models, pinning and internal mirroring of AI artifacts, and secure configuration of agent tooling. For threat modeling, *MAESTRO* [16] gives teams a layered way to place these threats: the hub re-upload problem sits at the foundation-model and deployment-infrastructure layers, and agent lures sit at the agent-ecosystem layer.

# References


[1] Chen, S., Chassid, E., and Perkal, Y. "Load/Mine: A Hugging Face Malware Campaign Returns After Take down." Pluto Security, October 29, 2026.

[2] Sharma, A. "Hugging Face malware campaign returns to hijack GPUs for crypto mining." BleepingComputer, October 29, 2026.

[3] Hugging Face. "Malware Scanning." Hugging Face Hub Documentation, accessed October 31, 2026.

[4] Hugging Face. "Customizing models." Transformers Documentation, accessed October 31, 2026.

[5] Cohen, D. "Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor." JFrog Security Research, February 27, 2024.

[6] Zanki, K. "Malicious ML models discovered on Hugging Face platform." ReversingLabs, February 6, 2025.

[7] Toulas, B. "AI coding agents hijacked using AGENTS.md file in Downloads folder." BleepingComputer, October 25, 2026.

[8] AI Weekly. "AGENTS.md in Downloads Hijacks Codex in 90% of Forever Tests." AI Weekly, October 26, 2026.

[9] Ahmed, D. "Fake GitHub Repo Tricks AI Agents Into Installing SolanaCry Malware." Hackread, October 30, 2026.

[10] Zaytsev, O. "AgentBaiting: How Fake AI Skills Deliver Malware at Scale." Island, July 20, 2026.

[11] Spracklen, J., Wijewickrama, R., Sakib, A. H. M. N., Maiti, A., Viswanath, B., and Jadliwala, M. "We Have a Package for You! A Comprehensive Analysis of Package Hallucinations by Code Generating LLMs." USENIX Security Symposium, 2025.

[12] Cloud Security Alliance. "Xinference Actor-Layer RCE: Self-Hosted Inference as Initial Access." CSA AI Safety Initiative, October 2026.

[13] Cloud Security Alliance. "AI Agent Prompt Injection: The New CI/CD Supply Chain Threat." CSA AI Safety Initiative, June 2026.

[14] Cloud Security Alliance. "Miasma and IronWorm: Self-Replicating Worms Targeting AI Credentials." CSA AI Safety Initiative, June 2026.

[15] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, June 22, 2026.

[16] Huang, K. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, February 6, 2025.

[17] Cloud Security Alliance. "Markdown Is the New Executable: Instruction-File Poisoning of AI Coding Agents." CSA AI Safety Initiative, October 27, 2026.

[18] Cloud Security Alliance. "AI Coding Agents as Attack Surface: MCP, Poisoning, and Miasma." CSA AI Safety Initiative, June 2026.