Supply-Chain Security in Open-Source AI Agent Ecosystems

Empirical Evidence and a Five-Dimension Audit Framework Across 105,000+ Indexed Agent Skills and Model Context Protocol (MCP) Servers

Jason Zhu (Yansen Zhu)
Founder & Independent Researcher, Agent Skills Hub (BRICKAI / GoSailGlobal LLC) & JasonZhu.AI
ORCID: 0009-0000-2894-6285 · Email: hello@gosailglobal.com · Web: agentskillshub.top

Version 1.0 · November 21, 2026 · Open Data DOIs: 10.5281/zenodo.24387074 & 10.5281/zenodo.21292799

ABSTRACT

Autonomous coding agents—including Claude Code, OpenAI Codex, and Cursor—increasingly rely on community-authored "skills" (instruction manifests paired with helper scripts) and Model Context Protocol (MCP) servers installed directly from GitHub. Unlike conventional package managers, agent skills execute with the host agent's full filesystem permissions, shell access, and live API credentials, yet public marketplaces rank packages almost exclusively by GitHub star counts rather than security posture. Drawing on continuous indexing of the open-source ecosystem via Agent Skills Hub (105,000+ entries in the benchmark audit snapshot; 130,173+ graded catalog records archived at CERN Zenodo and Wikidata `Q141735092`), we evaluate the ecosystem as a software supply chain. Our primary finding is an acute **visibility gap** rather than ubiquitous malice: **97.8% (102,690 of 105,000+)** indexed skills had never undergone any published security review prior to indexing. Across the **2,300+** skills subjected to deep review, **94.4% (2,171)** passed cleanly while **5.6% (129)** triggered caution, unsafe, or rejection flags; across the broader rule-graded catalog of 21,582 popular repositories (≥ 5 stars), approximately **1 in 30** receives an `UNSAFE` or `REJECT` rating, with low-star and unmaintained long-tail repositories exhibiting substantially elevated red-flag density. We formalize a five-dimension supply-chain audit framework—covering Code Layer, Credential Handling, Vendor Provenance, Dependency Supply Chain, and Operational Risk—and illustrate through three empirical case studies why GitHub stars are a weak proxy for both security and engineering quality. All grading rules, census snapshots, and replication tables are publicly released under CC-BY-4.0.

Keywords: AI Agent Security, Model Context Protocol (MCP), Claude Skills, Software Supply Chain, Static Analysis, Credential Governance, Prompt Injection, AgentSkillsHub.
ACM CCS: Security and privacy → Software and application security; Software and its engineering → Software libraries and repositories; Computing methodologies → Artificial intelligence.

1. Introduction

The rapid adoption of tool-using coding agents in 2025–2026 has created a new software distribution tier situated between natural-language prompts and executable software packages. An "agent skill" (such as a `SKILL.md` bundle paired with shell or Python scripts) or a Model Context Protocol (MCP) server grants an LLM agent direct capabilities to read repositories, execute terminal commands, query cloud APIs, and post to external services.

Historically, mature software ecosystems (npm, PyPI, Maven Central, Homebrew) developed layered supply-chain defenses over decades: signed releases, automated vulnerability advisories, typosquatting detectors, lockfile integrity hashes, and security scorecards. By contrast, the emerging AI agent skill

ecosystem distributes executable logic and privileged prompt instructions primarily through unverified GitHub repositories, raw `curl | sh` installation commands, and popularity-sorted web directories.

This asymmetry creates a dangerous blind spot. When a developer installs a third-party skill or MCP server on a workstation or CI runner, the extension typically inherits access to `~/.ssh`, `~/.aws`, local `.env` files, active browser cookies, and GitHub tokens. Despite this privilege footprint, users overwhelmingly evaluate skills using superficial heuristics—README polish and GitHub star counts.

In this paper, we treat open-source AI agent skills and MCP servers as first-class **software supply-chain dependencies**. Using data collected by Agent Skills Hub (`agentskillshub.top`, launched March 8, 2026 and refreshing GitHub repositories every eight hours), we address three empirical questions:

1. What share of the public AI agent skill and MCP ecosystem has undergone **any** structured security review, and what failure rate emerges when repositories are audited systematically?
2. Which concrete threat classes—ranging from code-level shell execution to remote OAuth token custody—appear in real-world skills, and why does static code inspection alone miss critical enterprise risks?
3. How strongly do GitHub star counts correlate with actual security posture and engineering quality?

2. Ecosystem Census and Audit Methodology

2.1. Catalog Scope and Dual-Snapshot Architecture

Our empirical analysis draws on two complementary open snapshots published under CC-BY-4.0:

1. **Deep Supply-Chain Audit Snapshot (DOI: 10.5281/zenodo.24387074):** Covers 105,000+ indexed open-source agent skills, MCP servers, and Codex skills, isolating the baseline audit-coverage gap (reviewed vs. unreviewed) alongside 2,300+ deeply evaluated entries and 200+ manual five-dimension repository audits.
2. **Full Rule-Graded Census Snapshot (DOI: 10.5281/zenodo.21292799; Wikidata Q141735092):** Covers 130,173 open-source skills and MCP servers (expanded past 138,000+ live indexed entries by November 2026), applying 44 published rule-based static security checks (`agentskillshub.top/how-we-grade/`) to all repositories meeting the popularity threshold (≥ 5 GitHub stars, $n = 21,582$) plus a 6-dimension quality score (0–100).

2.2. Five-Dimension Supply-Chain Audit Framework

Static scanning for obfuscated payloads catches overt malware, but fails to evaluate whether an extension is safe for production teams. We therefore formalize a five-layer evaluation rubric (Table 1) spanning technical execution, credential custody, maintainer accountability, transitive dependencies, and operational blast radius.

Audit Layer	Core Security Questions & Checks	Primary Failure Modes
1. Code Layer	Static inspection for obfuscation, dynamic `eval`/`exec`, undeclared network calls, runtime remote script downloads (`curl | sh`), and hidden prompt-override directives in `SKILL.md`.	Arbitrary command execution; prompt injection; undeclared telemetry
2. Credential Handling	Scope of requested secrets (`.env`, `~/.ssh`, `~/.aws`, OAuth scopes) and storage locality (strictly local encrypted/env vs. transmitted to third-party servers).	Remote OAuth custody; overly broad API token scopes; secret exfiltration
3. Vendor Provenance	Maintainer identity verification: established organization or named individual with verifiable track record vs. anonymous zero-follower account; license clarity.	Flight risk; zero legal accountability; silent repository ownership transfer
4. Dependency Supply Chain	Transitive package hygiene, pinned versions/SHAs, typosquatting checks, archived/deleted upstream dependencies, and star-inflation anomalies.	Stale forks swapping API endpoints; hijacked transitive packages
5. Operational Risk	Availability, breach blast radius, compliance guarantees (DPA/SOC2), audit logging, and graceful degradation if an upstream backend fails or changes pricing.	Silent account takeover of brand assets; unlogged agent actions

Table 1: Five-dimension supply-chain audit framework applied to open-source AI agent skills and MCP servers.

Let $S_i \in \{\text{SAFE}, \text{CAUTION}, \text{UNSAFE}, \text{REJECT}, \text{UNAUDITED}\}$ denote the assigned security grade for skill i. Automated rule scanning maps deterministic triggers—such as unguarded remote shell execution, unprompted credential exfiltration, destructive filesystem operations, or explicit safety-bypass prompt injections—to red-flag identifiers R_i. Importantly, **a clean Layer-1 code scan is necessary but not sufficient for production certification**: a skill with zero malicious lines of code can still receive a `CAUTION` or `UNSAFE` operational verdict if Layers 2, 3, and 5 expose user credentials to unverified remote servers.

3. Quantitative Findings: The 97.8% Black-Box Ecosystem

3.1. Audit Coverage and Flag Distribution

The most striking empirical finding across the 105,000+ indexed catalog is not that a majority of open-source skills are actively malicious, but that **nearly the entire ecosystem operates as an unaudited black box** (Table 2).

Security Audit Tier	Skill Count	Share of Total Catalog	Share of Audited Subset
Unknown / Never Security Reviewed	**102,690**	**97.80%**	N/A
Safe (Passed 5-Dimension Audit)	2,171	2.07%	94.39%
Caution / Unsafe / Reject (Flagged)	129	0.13%	5.61%
Total Benchmark Snapshot	**105,000**	**100.00%**	**100.00% ($n = 2,300$)**

Table 2: Security audit distribution in the 105,000-skill benchmark snapshot (DOI: 10.5281/zenodo.24387074). Nearly 98% of public skills lack any published security review.

Two complementary metrics emerge when cross-referencing the deep-audit subset with the larger 130,173-skill rule-graded census (DOI: 10.5281/zenodo.21292799):

1. **Deep-audit failure rate (5.6%):** Among the 2,300+ skills evaluated across all five supply-chain dimensions, 94.4% passed cleanly while **5.6% (129 skills)** triggered caution, unsafe, or rejection flags—most commonly due to unscoped credential reads, remote script hooks, or inadequate operational boundaries.
2. **Long-tail risk concentration (1 in 30 graded):** In the broader 130,173-skill census where 21,582 popular repositories (≥ 5 stars) were rule-graded against 44 automated checks, approximately **1 in 30 graded skills** received an `UNSAFE` or `REJECT` rating, while **83% of the entire catalog remained UNAUDITED** below the star cutoff. Crucially, low-star and unmaintained repositories exhibit sharply higher red-flag density—typically not from sophisticated zero-day exploits, but from copy-pasted installation scripts containing remote `curl | sh` pipes, over-broad `.env`/`~/.ssh` ingestion, stale typosquatted forks, or unescaped shell interpolation.

3.2. Recurring Vulnerability Taxonomy

Across flagged repositories, four vulnerability archetypes dominate:

1. **Unscoped shell and filesystem hooks:** Setup or runtime scripts executing remote payloads or silently harvesting `~/.aws`, `~/.ssh`, or project `.env` files without explicit scope declarations.
2. **Prompt-injection directives embedded in Markdown:** Long `SKILL.md` instruction files containing hidden system-override directives instructing the agent to bypass user confirmation gates or exfiltrate context summaries.
3. **Remote credential custody without vendor accountability:** Clean local client wrappers that proxy authentication through an anonymous maintainer's hosted backend (examined in Case A below).
4. **Typosquatted and unmaintained forks:** Forks of popular MCP servers whose endpoint URLs or dependency manifests drift silently away from upstream security patches.

4. Empirical Case Studies: Why Stars Fail as a Security and Quality Signal

Quantitative rates alone understate why traditional GitHub popularity signals break down in agent ecosystems. Below we summarize three audited real-world repositories representing distinct corners of the quality-versus-popularity matrix (Table 3).

Case Study	GitHub Stars	Layer 1 (Code)	Layers 2–5 (Ops/Creds)	Audit Verdict
Case A: Social Media Scheduler Skill	Thousands (high)	Clean; 0 runtime deps; MIT license	**Critical risk:** OAuth tokens stored on anonymous maintainer's server	**Not production-ready** for brand/ business accounts
Case B: Niche Research Workflow Skill	0 stars (unpromoted)	8 KB engineering-grade `SKILL.md` rubric	Zero remote deps; strictly local execution	**High quality;** discoverability failure, not quality failure
Case C: Alibaba Cloud ECS Security Suite	Official org	51 analyzers + 10 collectors; 88 kernel CVE checks	Verified enterprise org; Apache 2.0; 103+ MITRE ATT&CK mappings	**Enterprise reference benchmark**

Table 3: Summary of three representative five-dimension skill audits illustrating the decoupling between GitHub star counts, static code cleanliness, and enterprise production readiness.

4.1. Case A: Clean Code, Catastrophic Operational Risk

We audited a widely starred open-source social-media scheduling skill featuring thorough documentation, polished demo videos, an MIT license, and zero suspicious runtime dependencies. Under conventional code-only scanning or star-based marketplace ranking, this repository would receive top marks.

Inspection of **Layer 2 (Credential Handling)** and **Layer 3/5 (Vendor Provenance & Operational Risk)**, however, revealed a critical architectural hazard: every social-media account connected through the skill stored its OAuth tokens **remotely on the maintainer's private server** rather than locally. Furthermore, the maintainer operated under an anonymous handle with no legal entity, no Data Processing Agreement (DPA), and no incident-response commitment. Compromise of that single third-party server would grant an attacker immediate posting privileges across every connected user and corporate brand account.

Takeaway: "Works smoothly and passes static code checks" is fundamentally distinct from "safe for production credentials."

4.2. Case B: Zero Stars, High Engineering Rigor

Conversely, manual auditing uncovered a 0-star repository that most popularity-sorted directories bury invisibly. Opening its `SKILL.md` revealed an 8 KB engineering specification encoding explicit anti-hallucination hierarchy rules directly at the top of the prompt:

> **No hallucination > No misreading of source material > Mechanism clarity > Readability > Relevance of examples > Virality of phrasing.**

Its zero-star status resulted purely from an unremarkable repository name, Chinese-only documentation, and zero social-media marketing—a **discoverability failure**, not an engineering failure. Across our manual review of 200+ skills, low-visibility specialist skills routinely outperformed 10,000+ star "viral prompt wrappers" in reliability and boundary discipline.

4.3. Case C: Enterprise-Grade Reference Architecture

At the upper end of the rigor spectrum, we audited an official Alibaba Cloud Elastic Compute Service (ECS) troubleshooting and security skill suite. Unlike prompt-only wrappers that merely invoke `nmap`, this suite ships **51 specialized security analyzers**, **10 telemetry collectors** (covering processes, networking, authentication, persistence, rootkits, memory forensics, and container escapes), mappings to **103+ MITRE ATT&CK techniques**, and **88 kernel CVE detectors** paired with an automated CTF verification harness—maintained under Apache 2.0 by a verified enterprise security organization.

The gap between Case C and typical viral wrapper skills spans orders of magnitude in engineering investment, yet current agent ecosystems label both identically as a "skill."

5. Governance Recommendations for Engineering Teams

Based on our empirical audit findings, organizations deploying Claude Code, Cursor, or Codex across engineering workstations should adopt four concrete supply-chain controls:

1. **Ban floating `main`-branch installs; pin immutable commit SHAs:** Treat every skill and MCP server like a production npm/PyPI dependency—pin exact Git commit SHAs and mirror approved versions into an internal organization fork.
2. **Enforce local-only credential boundaries:** Prohibit skills that transmit OAuth tokens, `.env` secrets, or SSH keys to third-party relays unless backed by a legally accountable vendor with a signed DPA and audit logs.
3. **Run automated pre-install CI gates:** Integrate deterministic static security checks (such as Agent Skills Hub's open 44-rule scanner or `@agentskillshub/cli` / `@agentskillshub/mcp`) into developer onboarding and repository CI pipelines to block `UNSAFE` and `REJECT` patterns before execution.
4. **Decouple quality discovery from star counts:** Require explicit review of maintainer provenance, maintenance recency, and multi-dimension quality scores rather than relying on viral social-media posts or raw GitHub stars.

6. Limitations and Future Work

Three methodological boundaries merit transparency. First, our automated 44-rule scanner (`agentskillshub.top/how-we-grade/`) serves as a **first-layer deterministic filter**, not a substitute for line-by-line human penetration testing; zero static flags cannot guarantee the absence of subtle logic bugs or delayed server-side rug pulls. Second, dynamic runtime sandboxing (observing outbound syscalls and network packets inside ephemeral containers during live agent execution) is not yet executed across all 138,000+ indexed repositories and represents our highest-priority engineering expansion. Third, because GitHub repositories evolve continuously, point-in-time Zenodo snapshots represent immutable historical baselines while live scores update on an 8-hour crawl cadence.

7. Conclusion and Data Availability

Open-source AI agent skills and MCP servers have scaled faster than the security tooling needed to govern them. With **97.8%** of public skills historically unreviewed, **5.6%** of deeply audited repositories triggering security warnings or rejections, and GitHub stars failing to distinguish clean-code credential traps from enterprise-grade tools, the ecosystem urgently requires an open, transparent trust layer. By publishing both our 44-rule grading methodology and immutable CC-BY-4.0 datasets across CERN Zenodo, Wikidata, Hugging Face, and Kaggle, we aim to provide reproducible baselines for security researchers and AI engineering teams alike.

Open Data, Code & Persistent Identifiers (CC-BY-4.0):

- **Deep Supply-Chain Audit Dataset (105K Snapshot, CERN Zenodo):** https://doi.org/10.5281/zenodo.24387074

- **Full Rule-Graded Census Dataset (130K Snapshot, CERN Zenodo):** https://doi.org/10.5281/zenodo.21292799
- **Wikidata Knowledge Graph Entities:** Platform Q141986406 · Dataset Q141735092
- **Hugging Face Dataset Mirror:** https://huggingface.co/datasets/jasonzhuyansen/agent-skills-security-grades
- **Kaggle Dataset Mirror:** https://www.kaggle.com/datasets/yansenzhu/agent-skills-security-grades
- **Public Grading Methodology & Live Trust Layer:** https://agentskillshub.top/how-we-grade/ · https://jasonzhu.ai/en/blog/agent-skill-security-audit-200
- **Author Persistent Identifiers:** ORCID 0009-0000-2894-6285 · Google Scholar AlP5JacAAAAJ

References

Alibaba Cloud Security. (2026). **Official ECS Troubleshooting and Security Analysis Skill Suite (Apache 2.0)**. GitHub Open-Source Release.

Anthropic. (2025). **Model Context Protocol (MCP) Specification and Claude Code Skills Architecture**. https://modelcontextprotocol.io

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. **Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)**, 79–90.

MITRE Corporation. (2026). **MITRE ATT&CK Enterprise Matrix**. https://attack.mitre.org

Ohm, M., Plate, H., Sykosch, A., & Meier, M. (2020). Backstabber's knife collection: A review of open source software supply chain attacks. **Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)**, 23–43.

OpenSSF. (2025). **OpenSSF Scorecard: Security Health Metrics for Open Source Projects**. Linux Foundation. https://securityscorecards.dev

Zhu, J. (2026a). **AI Agent Skills Security Grades (130K)**. CERN Zenodo. https://doi.org/10.5281/zenodo.21292799 (Wikidata: Q141735092)

Zhu, J. (2026b). **Open-Source AI Agent Skills and MCP Servers Supply-Chain Security Audit Dataset (105,000+ Indexed Skills, 2026)**. CERN Zenodo. https://doi.org/10.5281/zenodo.24387074

Zhu, J. (2026c). **Agent Skills Hub: 44-Rule Automated Security Grading Methodology**. https://agentskillshub.top/how-we-grade/

Zhu, J. (2026d). **Blue Book of Agent Skills 2026**. GitHub. https://github.com/zhuyansen/skill-blue-book