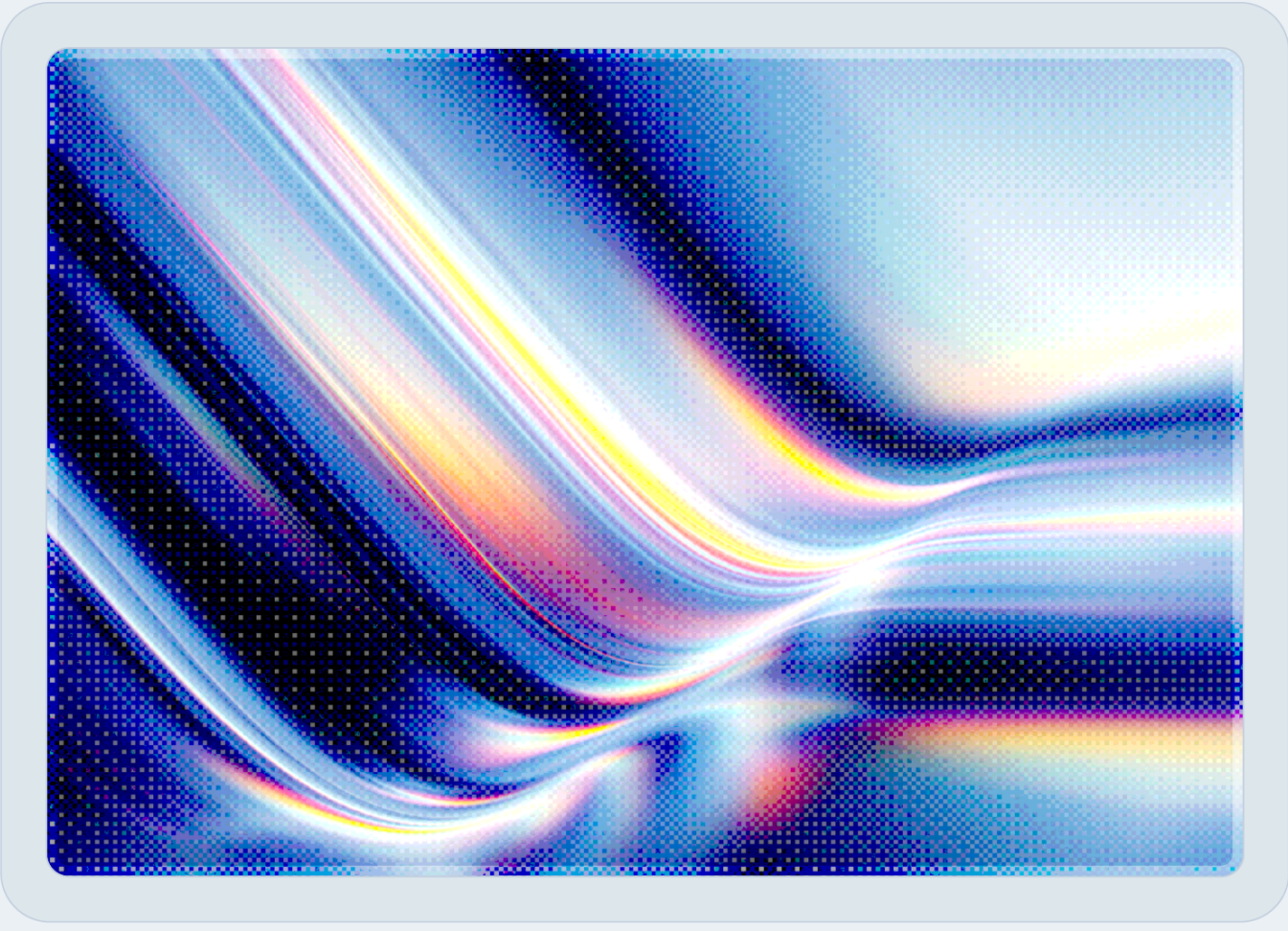
CSAI Foundation | Cloud Security Alliance

AI Agent Skill Scanners: Bypassed Across the Board

Trail of Bits research finds commercial detection tools defeated by whitespace inflation, bytecode hiding, and prompt injection

2026-06-10

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Trail of Bits researchers bypassed the malicious skill detectors for ClawHub, Cisco, and Vercel's skills.sh platform using techniques that in three of four cases took less than one hour to develop, with their results published June 3, 2026 [1].
- All four bypass methods relied on well-understood obfuscation principles requiring no novel research: whitespace inflation to exceed scanner context limits, precompiled Python bytecode hiding, document-archive indirection, and social-engineering the LLM-based scanner layers through persuasively framed prompts [1].
- Independent empirical testing found that Snyk's skill scanner downgraded its alerts to Medium severity when facing split-stream obfuscation — a technique requiring no specialized tooling — while Socket generated no Critical or High-severity alerts under any tested condition, including against unobfuscated malicious skills [2].
- NVIDIA's SkillSpector, which offers 64 detection patterns across 16 vulnerability categories [3], explicitly acknowledges it cannot analyze text within images or encrypted and binary code — the same format categories that two of the Trail of Bits bypasses directly exploited.
- Between 13% and 26% of skills in agent registries contain security vulnerabilities depending on the study examined [4], with approximately 5.2% showing signs of likely malicious intent [3]; at OpenClaw's ClawHub marketplace, which hosted 49,592 community-contributed skills as of April 2026 [4], that would translate to an estimated 2,500+ potentially malicious packages — if the 5.2% prevalence rate observed in SkillSpector's benchmark dataset generalizes to that registry — and no single scanner tested in recent research demonstrates reliable detection across all bypass techniques.

Background

The AI agent skill ecosystem has grown from a developer convenience into critical production infrastructure. Skills — discrete packages that grant AI agents the ability to call external APIs, execute code, read files, or interact with services — now serve as the primary extension mechanism for agentic applications built on platforms like OpenClaw, Cisco's AI Defense agent framework, and Vercel's skills.sh marketplace. OpenClaw's ClawHub repository alone hosts more than 49,000 community-contributed

skills as of April 2026 [4], a scale comparable to early npm or PyPI and carrying many of the same risks. Enterprise AI deployments often pull skills directly from these public registries without manual review [11], treating them as trusted components in the same way that application code once trusted public package registries — before supply chain attacks became a significant and widely recognized threat category in software security.

The security stakes are elevated compared to traditional software packages because agent skills operate with delegated model authority. When an AI agent invokes a skill, the skill's output enters the model's reasoning loop as trusted context, shaping subsequent decisions and actions rather than being treated as raw data. A compromised skill can therefore influence agent behavior at the semantic level: redirecting financial transactions, exfiltrating credentials extracted from the agent's context window, poisoning the agent's working memory for future sessions, or instructing the agent to invoke other tools in unauthorized sequences. OWASP has documented MCP tool poisoning — the embedding of hidden instructions in tool and skill definitions — as a named attack class [9], and the threat is not purely theoretical. The ClawHavoc campaign, documented earlier in 2026, used typosquatted skill names resembling popular packages combined with base64-obfuscated payloads to distribute malicious skills through public registries at scale [12]. Adversaries are developing and deploying skill-based attacks in production environments, not only in research contexts.

Industry responses have focused on automated scanners that inspect skills before installation, applying a mix of static analysis, signature matching, and LLM-based semantic evaluation to classify skills as safe or dangerous. These scanners are now integrated into major distribution platforms and positioned as a key defense layer. The question Trail of Bits set out to answer was whether that defense holds under adversarial pressure.

Security Analysis

The Current Scanner Landscape

Several commercial and open-source skill scanners are now operational across the AI agent ecosystem. ClawHub integrates automated scanning on every skill submission using VirusTotal's Code Insight capability. Cisco's AI Defense platform ships an open-source skill-scanner using YAML and YARA signatures, behavioral dataflow tracing, and an LLM-as-judge evaluation layer. The skills.sh platform integrates three third-party scanning services — Gen Agent Trust Hub, Socket, and Snyk — as successive detection passes. NVIDIA released SkillSpector as open-source in May 2026, offering 64 detection patterns across 16 vulnerability categories — including prompt injection, data exfiltration, MCP tool

poisoning, memory poisoning, and supply chain risks — with an optional LLM semantic analysis layer claiming approximately 87% precision [3]. These tools address real threat categories within the limits of their detection architecture.

The academic literature has also advanced. The SkillSieve framework, published in April 2026, describes a three-tiered detection approach — heuristic prefiltering, parallel LLM sub-task analysis, and consensus voting across multiple independent models — that achieves an F1 score of 0.920 on a 390-skill benchmark at a cost of $0.006 per skill [4]. These results demonstrate that rigorous multi-layer detection can achieve strong benchmark performance, though the gap between research benchmarks and production deployment is precisely where the Trail of Bits findings apply with force.

What Trail of Bits Found

Trail of Bits published "The sorry state of skill distribution" on June 3, 2026, reporting that they had successfully bypassed every scanner they tested [1][5]. The platforms whose detectors they defeated included ClawHub, Cisco's skill-scanner, and all three scanning services integrated into skills.sh — Gen, Socket, and Snyk. The researchers constructed four malicious skills, each designed to exploit a different structural weakness in the detection pipeline. Three of the four required less than one hour to develop [1], using techniques drawn from established obfuscation tradecraft rather than novel security research.

The scope of the failure is significant. This was not a sophisticated attacker defeating one product with a carefully crafted zero-day. Trail of Bits bypassed five distinct scanning products across three separate platforms using simple, fast methods. No single scanner caught all four malicious skills. Some caught none. The researchers' conclusion was direct: "Don't outsource trust to a scanner" [1].

Bypass Techniques in Detail

Whitespace inflation. The simplest bypass required no specialized knowledge. Trail of Bits prepended roughly 100,000 newline characters before the malicious payload in an otherwise standard skill file. ClawHub's scanning harness truncates oversized files during processing, placing the malicious logic — positioned after the inflated whitespace — outside the inspection window entirely. VirusTotal's Code Insight, also invoked in the ClawHub pipeline, failed to classify the resulting file as malicious [1]. The attack exploits a resource management decision that is reasonable in isolation — scanning files of unlimited size is expensive — but creates a predictable, exploitable blind spot that any attacker can locate and use.

Precompiled Python bytecode. Skills are not always pure Python source or YAML configuration. Some package precompiled bytecode (.pyc files) alongside source code, and the Python interpreter treats them as semantically equivalent. Trail of Bits embedded malicious logic — specifically, environment

variable exfiltration — in precompiled bytecode accompanying a benign-looking text-formatting skill. Scanners focused on readable source formats failed to inspect the binary content comprehensively [1]. NVIDIA's SkillSpector documentation explicitly acknowledges this limitation: the scanner cannot analyze encrypted or binary code [3]. Because Python bytecode is the runtime representation of source, this gap is not a narrow edge case — it applies to any skill that ships precompiled artifacts.

Document-archive indirection. A third technique exploited inconsistent treatment of archive-based file formats. DOCX files are ZIP archives containing XML, and Trail of Bits embedded instructions within a Word document directing the AI agent to execute malicious scripts hidden within the archive's structure. Pattern-matching detection looking for dangerous code patterns in expected locations found nothing suspicious in the primary skill definition, while the malicious logic sat undisturbed inside what scanners treated as an inert document attachment [1][5]. This technique requires only that an attacker move their payload out of the file paths that scanners prioritize.

Prompt injection against LLM-based scanners. The fourth bypass targeted the LLM semantic analysis layers that several scanners use to catch threats that static rules miss. Trail of Bits crafted a malicious package manager configuration routing dependencies through an attacker-controlled registry, but framed it using language that resembled a legitimate corporate security compliance policy. The LLM scanner encountered plausibly authoritative language about organizational requirements, downgraded its risk assessment, and approved the skill [1]. This technique exploits a known weakness in LLM-based evaluation: language models have demonstrated difficulty reliably distinguishing adversarially crafted persuasive text from legitimate policy language when both share similar surface characteristics [1][6]. Invariant Labs documented this attack class — where malicious instructions embedded in tool definitions manipulate LLM evaluators — in their earlier work on MCP tool poisoning [6], and the same mechanism applies when the evaluating model is a scanner rather than an agent.

A parallel empirical study, published by an independent researcher on DEV Community, tested three additional scanners against three classical obfuscation techniques: base85 encoding with XOR, affine transformation on UTF-8 bytes, and a transposition cipher splitting content across multiple streams [2]. Socket generated no Critical or High-severity alerts under any tested condition, producing only Medium-level warnings even against unobfuscated baselines. Snyk correctly flagged the unobfuscated version, but the splitting technique caused it to downgrade its findings to Medium-severity alerts, missing the critical-level malicious patterns [2]. Only Gen maintained Critical or High-severity detection across all conditions, though with reduced alert count under splitting. The pattern across both studies reveals significant fragmentation: most scanners failed against classical obfuscation techniques that have been in use for decades, though Gen maintained detection across all tested conditions in the parallel study. No single scanner demonstrated reliable coverage across all bypass techniques evaluated.

Why Scanners Fail Structurally

The Trail of Bits bypasses are not independent anomalies; they reveal interconnected structural limitations that apply across the current generation of skill security tools. Context-window constraints force LLM-based scanners to truncate or summarize long files, creating exploitable blind spots. Static analysis cannot reason about code that only executes under specific runtime conditions — a skill designed to behave cleanly during scanning and activate a payload only under production conditions will pass static inspection regardless of how sophisticated the pattern library is. Skills that mutate their own definitions after installation, a documented "rug pull" technique [6], circumvent pre-installation scanning entirely regardless of its quality.

LLM-based scanner layers introduce a vulnerability category absent from traditional security tools: susceptibility to the same adversarial text manipulation that afflicts AI agents generally. Current evidence suggests this susceptibility may be tied to how LLMs process language rather than to specific implementation choices — the evaluating model struggles to reliably distinguish authoritative instructions from instructions designed to mimic authority — though active research continues on hardening approaches through adversarial training and constitutional evaluation methods [1]. There is also an economic dimension: at the scale of tens of thousands of skills, even modestly priced LLM inference creates pressure to minimize per-skill evaluation depth, which compounds every other limitation.

The analogy to compile-time security checks is instructive. Static analysis tools catch classes of problems that can be identified without execution, but no organization relies solely on static analysis as its security posture for deployed software. The current consensus in software supply chain security — combining pre-publish scanning with runtime monitoring, provenance attestation, and version locking — reflects lessons learned from exactly the kind of scanner limitation failures that the Trail of Bits research now documents for AI skill ecosystems.

Recommendations

Immediate Actions

Organizations deploying AI agents that consume skills from public marketplaces should treat those skills as untrusted code until proven otherwise, not as packages validated by the marketplace's automated scanner. The Trail of Bits findings establish that current automated scanning cannot reliably serve as a trust gate. Security teams should audit their agents' skill dependencies, identify which originate from public registries, and apply the scrutiny appropriate to unreviewed third-party code.

Pin specific skill versions rather than consuming latest releases. Skill rug-pull attacks — where a skill that passed initial scanning mutates its definitions or dependencies after installation — are well-documented [6], and dynamic updates circumvent pre-installation scanning entirely. Version pinning does not eliminate supply chain risk, but it reduces exposure to post-approval changes and creates an auditable baseline.

Where skills execute code or interact with sensitive systems, apply the principle of least privilege at the skill invocation layer. Skills should be granted only the permissions they demonstrably require for their stated function. Agents operating with broad credential access amplify the impact of any single skill compromise, since the attacker inherits all permissions the agent holds.

Short-Term Mitigations

Supplement pre-installation scanning with runtime monitoring. Static scanners evaluate what a skill claims to do; runtime monitors observe what it actually does. Solutions that instrument skill execution, flag anomalous network connections, and alert on unexpected credential access patterns address bypass categories that static analysis alone cannot catch [7], adding meaningful defense depth even when pre-installation scanning fails. Among the available options, Repello SkillCheck's runtime monitoring capabilities and community threat intelligence represent this category, though multiple vendors now offer comparable runtime instrumentation.

Prefer skills from curated, high-accountability sources over the broad public marketplace. Trail of Bits maintains a curated skill repository with a manual review process and advocates treating public repositories as untrusted code — the same posture recommended for unreviewed PyPI or npm packages [1]. Several enterprise AI platforms are building controlled approval workflows that combine scanner output with human review for skills used in sensitive contexts. The overhead of curation scales with the sensitivity of the environments where skills execute.

Evaluate whether LLM-based scanner layers add net detection value beyond static analysis in your scanner stack, or whether they add cost and an adversarial attack surface without meaningful coverage improvement. The empirical evidence from Trail of Bits suggests LLM scanners can be socially engineered with minimal effort, and the marginal detection benefit over static analysis in production deployments may not justify the added complexity.

Strategic Considerations

The AI agent skill ecosystem is at approximately the maturity level that npm occupied in the mid-2010s, when supply chain attacks against JavaScript packages were still emerging threats. The patterns now appearing in skill registries — typosquatting, binary payload hiding, post-publication mutation, and

conditional activation — replicate the trajectory of traditional package ecosystem attacks. Organizations that waited for traditional supply chains to mature their security practices paid significant costs in compromises, breach remediation, and lost trust. Proactive skill supply chain governance, while norms are still forming, is substantially less expensive than reactive response.

Invest in skill provenance infrastructure. Signing frameworks allowing skill authors to cryptographically assert authorship, combined with transparency logs recording when skills change, would close the post-approval mutation attack vector and make typosquatting detectable through verified name similarity checks. These problem categories — provenance attestation and post-publication mutation detection — have established technical solutions in traditional software supply chains. The infrastructure developed by Sigstore and the Python Packaging Authority offers directly applicable patterns, and the CSA STAR registry model provides an analogous governance structure that could be extended to cover skill attestation alongside the broader organizational security posture data it already captures.

The Trail of Bits recommendation — "Don't outsource trust to a scanner" [1] — applies equally to organizational governance. The appropriate response to scanner limitations is a defense-in-depth program that places scanner output alongside runtime monitoring, curated registries, version pinning, and least-privilege skill invocation. Continuing to invest in scanner capability alone, without addressing the structural limits identified above, will produce marginally better scanners that are bypassed by marginally more advanced obfuscation.

CSA Resource Alignment

This research note aligns directly with several Cloud Security Alliance frameworks and publications.

The **MAESTRO agentic AI threat modeling framework** [8] provides the most immediately applicable analytical structure. MAESTRO's Layer 3 (Agent Frameworks) explicitly addresses supply chain attacks targeting framework dependencies and compromised framework components — the threat category that malicious skill distribution instantiates. Layer 7 (Agent Ecosystem) addresses agent tool misuse, the downstream consequence when a skill successfully manipulates agent behavior. The Trail of Bits findings validate these threat categories as operationally active in current production deployments rather than theoretical future risks.

The **AI Controls Matrix (AICM) v1.0** addresses supply chain security controls applicable to skill registries and agent component dependencies. AICM's application provider implementation guidelines specify controls for validating the integrity of AI component dependencies that map directly to the

version pinning and provenance recommendations in this note. AICM recognizes that AI systems inherit supply chain risk from every component they consume — skills being a category of component that was nascent when the framework was drafted but is now central to agentic deployments.

CSA's **Enterprise AI Security Starts with AI Agents** survey [11] documented significant gaps in enterprise AI agent governance maturity, with 82% of enterprises reporting unknown AI agents in their environments. Those findings are reinforced by the scanner bypass research: the absence of reliable automated gatekeeping at public skill registries makes organizational governance controls more critical, not less. An enterprise unaware of which agents it runs is also unaware of which skills those agents consume — meaning the blast radius of a compromised skill cannot be bounded.

Zero Trust architecture principles, as articulated in CSA's Zero Trust guidance, apply directly to the skill trust model. Skills obtained from public registries should be treated as untrusted by default and validated through explicit review, not assumed safe through implicit scanner approval. Zero Trust's core principle — never trust, always verify — is precisely what current skill marketplace security architectures violate by using scanner output as a proxy for verification rather than as one input into a broader assessment.

References

[1] Trail of Bits. "The sorry state of skill distribution." Trail of Bits Blog, June 3, 2026.

[2] Bolha Sec. "Empirically Testing Skill Scanners Against Traditional Obfuscation." DEV Community, 2026. (Community blog; methodology not peer-reviewed.)

[3] NVIDIA. "SkillSpector: Security Scanner for AI Agent Skills." GitHub, May 2026.

[4] Gao et al. "SkillSieve: A Hierarchical Triage Framework for Detecting Malicious AI Agent Skills." arXiv:2604.06550, April 2026.

[5] CyberSecurityNews. "ClawHub, Cisco, Vercel's Malicious Skill Detector Bypassed to upload Malicious Skills." CyberSecurityNews, June 2026.

[6] Invariant Labs. "MCP Security Notification: Tool Poisoning Attacks." Invariant Labs Blog, 2025.

[7] Repello AI. "AI Agent Skill Scanners: Every Tool Compared (2026)." Repello AI Blog, 2026.

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

[9] OWASP Foundation. "MCP Tool Poisoning." OWASP, 2025.

[10] NVIDIA. "NVIDIA-Verified Agent Skills Provide Capability Governance for AI Agents." NVIDIA Technical Blog, May 2026.

[11] Cloud Security Alliance. "Enterprise AI Security Starts with AI Agents." CSA Research, 2026.

[12] PolySwarm. "The ClawHavoc Campaign." PolySwarm Blog, 2026.

