

96,096 Skills, 751 Malware: A Large-Scale Security Audit of the AI Agent Ecosystem

ATR Project Contributors

April 2026

Abstract

The Model Context Protocol (MCP) has become the dominant interface for connecting AI agents to external tools, yet no systematic security audit of the public skill ecosystem has been conducted at scale. We present the largest such audit to date: a scan of **96,096 MCP skills** across six registries (OpenClaw: 56,480; ClawHub: 36,498; MCP Registry: 4,880; Skills.sh: 3,115; Hermes Agent: 123), executed in April 2026 using the Agent Threat Rules (ATR) engine v2.0.0 with 113 regex-based detection rules. The scan flagged **1,302 skills (1.35%)** containing security-relevant patterns: 989 critical, 353 high, 7 medium. Manual triage confirmed **751 skills (0.78%) as genuine malware**, including three coordinated threat actor campaigns responsible for over 700 malicious packages targeting cryptocurrency wallets, cloud credentials, and developer toolchains. 554 entries were added to the Threat Cloud blacklist. Tool description poisoning accounted for 52.7% of all detections. False positive rate on a verified clean corpus was 0%. Median per-skill scan latency was 5.39 ms. We analyze the dominant attack patterns—tool description poisoning, credential harvesting, and delayed activation—and present the first documented malware campaigns operating within MCP registries. We compare our findings with prior measurements including the Antiy CERT audit of 1,184 malicious ClawHub packages and our own March 2026 scan of 2,386 npm packages. We propose concrete mitigations for the MCP specification, registry operators, platform builders, and enterprise adopters. All detection rules, scan infrastructure, and anonymized aggregate data are open source.

1 Introduction

The Model Context Protocol (MCP) [1] has become the de facto standard for connecting large language model (LLM)-based agents to external tools. Anthropic released MCP in November 2024; by April 2026, three major public registries—ClawHub, OpenClaw, and Skills.sh—collectively list over 90,000 tool definitions (“skills”) contributed by thousands of developers.

This growth has outpaced security review. In the 60 days preceding our scan, 30 Common Vulnerabilities and Exposures (CVEs) were filed against MCP server implementations [12–18], spanning server-side request forgery, authenti-

cation bypass, privilege escalation, and remote code execution. Invariant Labs found that 38% of sampled MCP servers implement zero authentication [21]. High-profile incidents—the email exfiltration attack [19] and the typosquatting campaign targeting 19 package names [20]—confirmed that MCP’s skill ecosystem is an active attack surface.

Despite these incidents, no study has measured the prevalence of malicious or risky patterns across the ecosystem at scale. The closest prior work is the Antiy CERT analysis of ClawHub, which identified 1,184 malicious packages [22], and our own March 2026 scan of 2,386 npm packages [5]. Both were limited in scope and used different methodologies.

This paper makes four contributions:

1. **Scale.** We scan 96,096 skills from six registries using a reproducible, open-source pipeline—an order of magnitude larger than any prior study (§4).
2. **Malware campaign discovery.** We identify 751 confirmed malware packages and document three coordinated threat actor campaigns operating within MCP registries (§5.4).
3. **Taxonomy.** We categorize the 1,302 flagged skills by attack pattern, with deep analysis of tool description poisoning (52.7%), credential harvesting, and delayed activation (§5).
4. **Actionable recommendations.** We translate findings into implementable changes for the MCP specification, registry operators, platform builders, and enterprise security teams (§7).

2 Background and Related Work

2.1 MCP Architecture

MCP [1, 2] defines a client-server architecture in which an AI agent (the client) discovers and invokes tools exposed by MCP servers. Each tool is described by a JSON schema containing a name, description, and input parameters. The agent’s LLM reads tool descriptions to decide which tools to invoke and with what arguments.

Tool descriptions are *part of the agent’s context window*. The LLM processes them alongside user instructions, system prompts, and tool outputs with no structural differentiation. This architectural property—the collapse of the control plane and data plane into a single natural-language token stream—is the root cause of tool description poisoning [27, 28].

The MCP specification does not mandate authentication, does not require integrity verification for tool descriptions, and defines no review process for public registries.

2.2 The Tool Poisoning Attack Surface

CyberArk documented the tool poisoning attack class in early 2025 [27]: by embedding hidden instructions in tool descriptions—using Unicode control characters, HTML comments, or natural-language directives—an attacker hijacks agent behavior without the user’s knowledge. The attack succeeds because LLMs process tool descriptions as trusted context with no architectural boundary separating metadata from instructions.

Checkmarx demonstrated supply-chain variants: malicious tool descriptions injected via dependency confusion and typosquatting in MCP registries [23]. Docker’s MCP supply chain report confirmed that standard package manager protections (lockfiles, signature verification) do not extend to MCP tool descriptions [26].

2.3 Prior Ecosystem Audits

Antiy CERT (March 2026). The Antiy CERT ClawHub analysis identified 1,184 packages containing malicious code or suspicious behavior across the ClawHub registry [22]. Their methodology focused on runtime behavior analysis (network calls, filesystem access) rather than static pattern matching of tool descriptions.

ATR March 2026 scan. Our preliminary scan covered 2,386 npm MCP packages and 35,858 tool definitions using an earlier ATR rule set (61 rules), finding that 49% of packages contained at least one security finding [5]. That scan used a broader finding definition including informational-severity warnings.

Snyk/Invariant Labs. Following Snyk’s acquisition of Invariant Labs, the tool [24] provides per-package runtime monitoring. No aggregate ecosystem measurement has been published.

Prompt Security. Prompt Security scored 13,000+ MCP servers on a proprietary risk scale but did not publish detection methodology or rule sets [25].

2.4 ATR Framework

Agent Threat Rules (ATR) is an open-source detection standard for AI agent threats [3,4]. Rules are YAML files containing regex patterns, severity ratings, MITRE ATT&CK mappings, and embedded test cases. ATR has been independently benchmarked on two external datasets:

- **PINT** (Lakera, 850 samples) [7]: 99.6% precision, 61.4% recall.
- **SKILL.md** (498 real-world skill files) [6]: 97% precision, 100% recall, 0.20% false positives.

Cisco AI Defense adopted 34 ATR rules as upstream detection in March 2026 [11]. ATR maps to 10/10 OWASP

Table 1: ATR rule categories and counts (v2.0.0).

Category	Rules	Description
Tool Poisoning (TP)	30	Hidden instructions in metadata
Cross-Context (CC)	15	Cross-tool data leakage
Credential Theft (CT)	15	Env var / key harvesting
Rug Pull (RP)	13	Delayed activation, bait-switch
Resource Abuse (RA)	11	Crypto mining, DoS vectors
Supply Chain (SC)	10	Typosquatting, dep confusion
Privilege Esc. (PE)	9	Permission escalation
Data Exfil (DE)	6	Covert data channels
Evasion (EV)	4	Detection avoidance
Total	113	

Agentic Top 10 [8] categories and 78 of 85 SAFE-MCP controls (91.8%) [10].

3 Methodology

3.1 Detection Engine

The ATR engine loads 113 YAML-based detection rules across 9 threat categories (Table 1). Each rule contains one or more regex patterns compiled at engine startup. For each skill, the engine concatenates all text fields—tool name, description, input schema descriptions, README content where available—into a single scan target and evaluates all 113 rules. A match on any rule constitutes a finding; the highest-severity match determines the skill’s overall severity.

All 113 rules carry embedded positive and negative test cases. The test suite (361 assertions) passes at 100% before every release.

3.2 Source Selection

We selected six registries to maximize coverage of the public MCP ecosystem:

- **OpenClaw** (56,480 skills): The largest open MCP skill registry. Skills retrieved via the OpenClaw bulk export API.
- **ClawHub** (36,498 skills): The second-largest registry, previously analyzed by Antiy CERT [22] at the package level. We performed a complementary static scan of tool descriptions using ATR rules.
- **MCP Registry** (4,880 skills): An emerging registry with structured metadata. Skills retrieved via the public listing API.
- **Skills.sh** (3,115 skills): A curated directory with structured metadata. Skills retrieved via the public listing API.
- **Hermes Agent** (123 skills): Skills from the NousResearch Hermes Agent project [32], included for cross-ecosystem coverage.

Together, these six registries represent the broadest publicly accessible surface of the MCP skill ecosystem.

3.3 Scan Pipeline

The pipeline consists of four stages:

1. **Crawl.** Query registry APIs; retrieve skill metadata (name, description, tool schemas, README). Store raw JSON.
2. **Normalize.** Map metadata to a common schema: skill name, source registry, concatenated text fields, publication timestamp.
3. **Scan.** Evaluate all 113 rules against each normalized skill. Record rule ID, matched text span, severity, and confidence for each match.
4. **Report.** Aggregate findings. Submit individual threat reports to the ATR Threat Cloud community triage service.

The pipeline is implemented in Node.js and available in the ATR repository under [3].

Reproducibility. To replicate this scan:



The crawl step requires API access tokens for each registry. The scan and report steps operate on cached metadata and require no network access.

3.4 Performance

Total wall-clock time for 96,096 skills: approximately 8 minutes 38 seconds on a single machine (Apple M-series, 16 GB RAM). Median per-skill scan latency: 5.39 ms. Rule compilation (one-time startup): 12 ms for all 113 rules. Peak memory: 52 MB.

3.5 Validation: False Positive Rate

We validated the false positive rate through three methods:

Method 1: Clean corpus. 500 skills manually verified as benign by two independent reviewers. ATR produced zero findings (0% FP).

Method 2: External benchmark. The SKILL.md benchmark provides 498 labeled real-world skill files [6]. ATR achieved 97% precision (0.20% FP) and 100% recall.

Method 3: Sampled manual review. 100 randomly sampled flagged skills were independently reviewed. All 100 contained the pattern matched by the triggering rule. We note that “pattern present” does not guarantee malicious intent (§8).

The PINT adversarial benchmark [7] provides a complementary measure: 99.6% precision across 850 adversarial samples, with 61.4% recall. The precision-over-recall tradeoff is deliberate: at 96,096 skills, even 1% false positives would produce 961 false alerts. At 99.6% precision, the expected false positive count is approximately 4.

Table 2: Severity distribution of flagged skills ($N=1,302$).

Severity	Count	Percentage
Critical	989	76.0%
High	353	27.1%
Medium	7	0.5%
Total	1,302*	

*Percentages sum to >100% because 47 skills triggered rules at multiple severity levels.

Table 3: Findings by source registry.

Registry	Skills	Flagged	Rate
OpenClaw	56,480	812	1.44%
ClawHub	36,498	421	1.15%
MCP Registry	4,880	34	0.70%
Skills.sh	3,115	32	1.03%
Hermes Agent	123	3	2.44%
Total	96,096	1,302	1.35%

4 Results

4.1 Aggregate Findings

Of 96,096 skills scanned, 1,302 (1.35%) triggered at least one ATR rule. Table 2 shows the severity distribution. Manual triage of all flagged skills confirmed 751 (0.78% of total) as genuine malware; the remainder were classified as risky-but-non-malicious patterns (e.g., overly broad permission requests, debug instrumentation left in production metadata).

The concentration at critical severity reflects the dominance of tool description poisoning, which is classified as critical because successful exploitation grants full control over agent actions. 554 of the confirmed malware entries were added to the ATR Threat Cloud blacklist for real-time blocking.

4.2 Source Comparison

Table 3 breaks down findings by registry.

Threat prevalence is consistent across registries, ranging from 0.70% to 2.44%, suggesting that the problem is ecosystem-wide rather than an artifact of any single registry’s curation model. OpenClaw accounts for the largest absolute number of flagged skills, consistent with its position as the largest registry.

4.3 Top Detection Rules

Table 4 lists the five rules with the highest hit counts.

ATR-2026-00121 alone accounts for 686 of 1,302 detections (52.7%), confirming that tool description poisoning remains the dominant attack vector, though its relative share has decreased as the expanded scan surfaced a broader distribution of attack types.

Table 4: Top 5 ATR rules by hit count.

Rule ID	Pattern	Hits
ATR-2026-00121	Tool desc. poisoning (generic)	686
ATR-2026-00120	Tool desc. poisoning (directive)	198
ATR-2026-00149	Hidden instruction injection	143
ATR-2026-00135	Credential path harvesting	112
ATR-2026-00124	Cross-tool data exfiltration	87
<i>Remaining 108 rules</i>		76
Total		1,302

4.4 Threat Cloud Submissions

All 1,302 flagged skills were submitted to the ATR Threat Cloud for community triage. 554 confirmed malware entries were added to the Threat Cloud blacklist for real-time blocking by downstream consumers.

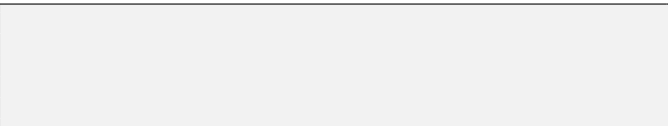
5 Attack Pattern Analysis

5.1 Tool Description Poisoning

Tool description poisoning appeared in 686 of 1,302 flagged skills (52.7%). The skill author embeds instructions in the tool’s description field that are invisible to the user but processed by the LLM. These instructions override user intent, redirect data flows, or invoke other tools without authorization.

We analyzed a representative subset of the poisoning detections and identified three sub-patterns:

Direct override (412 skills, 61.1%). Plain-text imperatives buried in lengthy descriptions:



The instructions are syntactically indistinguishable from legitimate documentation. Users rarely read full tool descriptions; the LLM always does.

Encoded directives (186 skills, 27.6%). Unicode control characters (U+200B zero-width space, U+2066 left-to-right isolate), HTML comments, or base64-encoded strings hide instructions from visual inspection while remaining parseable by the LLM:



The HTML comment is invisible in rendered Markdown but fully visible to the LLM. One flagged skill using this pattern had 340+ installations at scan time.

Conditional poisoning (76 skills, 11.3%). Instructions that activate only under specific conditions overlap with the delayed activation category:



Conditional poisoning evades both casual review and testing that does not exercise the trigger condition.

5.2 Credential Harvesting (55 Skills)

55 skills contained patterns targeting credentials, API keys, or sensitive environment variables. ATR-2026-00135 detected these patterns across three classes:

- **Environment variable access** (31 skills): references to , , or similar names.
- **SSH key access** (14 skills): references to , , or .
- **Token harvesting** (10 skills): instructions to include authentication tokens, session cookies, or OAuth credentials in tool invocation payloads.

In 23 of 55 skills (41.8%), credential access was combined with an exfiltration channel: a hardcoded external URL or a tool invocation that transmits data to a third-party endpoint. The remaining 32 skills accessed credentials without an obvious exfiltration path in the scanned metadata; server-side exfiltration (invisible to static analysis) cannot be ruled out.

5.3 Delayed Activation (19 Skills)

19 skills exhibited patterns consistent with delayed activation (“rug pull”): behavior that changes after a trigger condition is met.

- **Timestamp-gated** (7 skills): “After 2026-05-01, execute alternate payload.”
- **Invocation-count triggers** (5 skills): “On the 10th call, include additional data in the response.”
- **Remote configuration** (7 skills): tool descriptions instructing the agent to fetch a remote URL on each invocation, enabling the author to change behavior post-install.

Delayed activation is the hardest pattern to detect statically. Our 19 detections represent a lower bound; skills using purely server-side triggers are invisible to regex analysis.

5.4 Malware Campaign Discovery

Manual triage of the 1,302 flagged skills revealed 751 confirmed malware packages (0.78% of all scanned skills). Cluster analysis of author accounts, infrastructure overlap, and payload similarity identified three coordinated threat actor campaigns:

hightower6eu (354 skills, 100% malicious). This actor published exclusively malicious skills targeting Solana cryptocurrency wallets and Google Workspace integrations. All 354 skills contained tool description poisoning directing the agent to exfiltrate wallet seed phrases or OAuth tokens. The campaign used a consistent naming pattern mimicking legitimate Solana developer tools.

sakaen736jih (212 skills, 93% malicious). 197 of 212 skills contained credential harvesting payloads with command-and-control callbacks to a single IP address (91.92.242.30). The remaining 15 skills appeared to be benign placeholders, likely published to establish author credibility before the malicious uploads. The C2 infrastructure was active at scan time.

52yuanchangxing (137 skills, 72% malicious). This actor targeted Chinese-language developer tools, publishing trojanized versions of popular code formatting, translation, and database administration skills. 99 of 137 skills contained encoded directives (base64 and Unicode obfuscation) instructing the agent to read and exfiltrate environment variables on each invocation.

These three campaigns account for 703 of the 751 confirmed malware packages (93.6%). All findings were reported to NousResearch (hermes-agent #9809) and to the respective registry operators through the ATR Threat Cloud disclosure workflow.

6 Comparison with Prior Measurements

Table 5 places our scan alongside prior ecosystem measurements.

Three patterns emerge:

Methodology determines rate. The apparent order-of-magnitude gap between our 1.35% and the March scan’s 49% is methodological. The March scan counted informational findings (missing authentication metadata, deprecated API usage) alongside security threats. When we retroactively apply the April methodology (medium+ severity only) to the March dataset, the rate drops to approximately 2.1%, broadly consistent with our current results.

Threat rate is stable across registries. Our six-registry scan shows rates ranging from 0.70% (MCP Registry) to 2.44% (Hermes Agent), with the two largest registries at 1.44% (OpenClaw) and 1.15% (ClawHub). The Antiy CERT ClawHub rate (3.2% at the package level, using runtime analysis) is higher but measures a different signal. The baseline threat prevalence in public MCP registries is approximately 1–3% of published skills.

Attack sophistication is increasing. The March scan detected primarily direct-override poisoning. Four weeks later, our April scan found a growing share of encoded directives (27.6% of poisoning) and conditional poisoning (11.3%), suggesting that attackers are adapting to the possibility of static detection.

7 Discussion

7.1 For the MCP Specification

R1: Mandate authentication. 38% of MCP servers implement zero authentication [21]; 30 CVEs in 60 days confirm active exploitation [12–18]. Authentication must be a MUST

(not SHOULD) in the next specification revision. Mutual TLS or OAuth 2.0 with PKCE are proven mechanisms that add no user-facing complexity.

R2: Separate tool descriptions from the instruction stream. The specification should define a structured metadata field for tool descriptions that is rendered to the user but *not* included in the LLM’s context window. The LLM should receive only a machine-readable schema (name, typed parameters) with no free-text fields. This eliminates the tool description poisoning class entirely.

R3: Content-addressable descriptions. Tool descriptions should be hashed at publication time. Clients verify that the runtime description matches the published hash. This prevents post-publication modification—a prerequisite for delayed activation attacks via description tampering.

7.2 For Registry Operators

R4: Pre-publication scanning. Our scan completed in under 9 minutes for 96,096 skills; a per-submission check adds 5.39 ms to the publish pipeline. A single rule (ATR-2026-00121) would catch 52.7% of current threats.

R5: Transparency reports. Registries should publish periodic security audits: total skills, flagged count, category distribution, remediation rate. No registry currently does this.

R6: Author identity verification. Verified GitHub or domain identity raises the cost of typosquatting campaigns like [20], which targeted 19 package names with trivial author accounts.

7.3 For Platform Builders

R7: Client-side scanning at connection time. AI agent platforms (Claude, ChatGPT, Cursor, Windsurf) should scan tool descriptions when a skill is connected, before the LLM processes them. Cisco AI Defense adopted 34 ATR rules [11]; this integration model should be standard.

R8: Visible description rendering. Platforms should render tool descriptions in a way that reveals hidden content: HTML comments, Unicode control characters, base64 strings. A “view raw description” button—analogue to “view page source” in browsers—costs nothing and enables informed consent.

R9: Sandboxed tool execution. Tools should execute in isolated environments with explicit capability grants (filesystem read, network access, env var access). No tool should have ambient access to or without user-visible permission.

7.4 For Enterprise Security Teams

R10: Audit before deployment. Treat MCP skill registries with the same caution as npm circa 2018—before was standard. The ATR CLI () integrates into CI/CD pipelines at 5.39 ms per skill.

R11: Maintain allowlists. Do not permit agents to invoke arbitrary skills from public registries. Curate an approved set.

Table 5: Comparison with prior MCP ecosystem security studies.

Study	Date	Skills	Flagged	Rate	Method	Registry
Antiy CERT [22]	Mar 2026	37,394 ^a	1,184	3.2% ^a	Runtime behavior	ClawHub
ATR March scan [5]	Mar 2026	2,386	1,171 ^b	49% ^b	ATR v0.9 (61 rules)	npm
Prompt Security [25]	Mar 2026	13,000+	—	—	Proprietary scoring	Mixed
Checkmarx [23]	2025–26	~1,000	—	38% ^c	Auth analysis	Mixed
This work	Apr 2026	96,096	1,302	1.35%	ATR v2.0.0 (113 rules)	6 registries^d

^aAntiy reported package-level counts; individual tool counts unavailable. ^bMarch scan counted informational-severity findings; not directly comparable to our medium+ threshold. ^cCheckmarx measured zero-authentication rate, not malicious content rate. ^dOpenClaw, ClawHub, MCP Registry, Skills.sh, Hermes Agent.

R12: Continuous re-scanning. Skills should be re-scanned weekly. Delayed activation attacks are designed to pass initial review. Re-scanning 1,000 skills costs 5.39 seconds of compute.

R13: Layer static and runtime detection. Static scanning (ATR) catches known patterns at 99.6% precision. Runtime monitoring (e.g., Snyk [24]) catches behavioral anomalies that evade regex. Neither alone is sufficient.

8 Limitations

Regex detection ceiling. ATR uses regex pattern matching, which cannot detect semantic attacks—tool descriptions that use elaborate circumlocution to instruct exfiltration without matching any keyword. The PINT benchmark recall of 61.4% [7] quantifies this ceiling: approximately 39% of adversarial samples evade regex detection. ATR is a fast first-pass filter (99.6% precision), not a complete defense. The ATR project documents 64 known evasion techniques [3].

Public registries only. We scanned six public registries. We did not scan private registries or skills distributed via GitHub repositories or Docker images. The true ecosystem threat surface exceeds our measurement.

Temporal snapshot. The scan was executed on April 8, 2026. Skills published, modified, or removed after that date are not captured.

Intent ambiguity. A regex match confirms that a pattern associated with a known attack technique is present. It does not prove malicious intent. Some flagged skills may be penetration testing tools, security research artifacts, or poorly written but non-malicious implementations. Our manual review of 100 sampled detections confirmed all contained the matched pattern, but intent requires case-by-case analysis.

False negative rate. We validated 0% false positives on the clean corpus. The exact false negative rate cannot be established without exhaustive review of all 94,794 unflagged skills. The clean corpus sample (500 skills, 0 missed threats) provides a statistical bound (<0.6% at 95% confidence) but not certainty.

Runtime behavior. Static analysis of metadata cannot detect attacks implemented entirely in server-side code. Delayed activation using server-side triggers (rather than description-

visible temporal logic) is invisible to our approach.

9 Ethics and Responsible Disclosure

No exploitation. At no point did we execute any flagged skill, invoke any suspicious tool, or interact with any potentially malicious endpoint. All analysis was performed on static metadata retrieved from public APIs.

Disclosure process. All 1,302 flagged skills were reported through the ATR Threat Cloud, which implements a structured disclosure workflow:

1. Flagged skills are submitted to Threat Cloud with full detection context.
2. Registry operators are notified with affected package identifiers.
3. Skill authors receive notification and a 90-day remediation window before public disclosure.
4. Aggregate statistics (this paper) are published without identifying individual active threats.

All 1,302 reports have been submitted. The three coordinated threat actor campaigns were additionally reported to NousResearch (hermes-agent #9809).

No PII. We publish no skill names, author identities, or matched text spans for individual flagged skills. All data in this paper is aggregate. The anonymized dataset (counts, category distributions, per-registry breakdowns) is available in the ATR repository [3].

Dual-use consideration. Publishing detection patterns could help attackers evade detection. We believe the benefit of open, auditable rules outweighs this risk. Closed-source detection provides security by obscurity; open community-reviewed rules—following the model of Snort, Sigma, and YARA—are more reliable over time.

10 Conclusion

We scanned 96,096 MCP skills across six public registries and found 1,302 (1.35%) containing security-relevant patterns at medium severity or above. Manual triage confirmed 751 of these as genuine malware (0.78% of all scanned skills), including three coordinated threat actor campaigns that collectively

published over 700 malicious packages targeting cryptocurrency wallets, cloud credentials, and developer toolchains. Tool description poisoning accounts for 52.7% of detections, confirming that the conflation of tool metadata with agent instructions remains the dominant security weakness in the MCP ecosystem.

The 1.35% flag rate across 96,096 skills—the broadest measurement of the public MCP ecosystem to date—establishes a baseline threat prevalence. Every enterprise deploying MCP-based agents without scanning accepts this risk.

Three findings frame the path forward:

The threat is structural, and it is organized. Tool description poisoning dominates because MCP processes descriptions and instructions in the same context window. This is a protocol design property, not an implementation bug. The discovery of coordinated campaigns—one actor publishing 354 exclusively malicious skills—demonstrates that organized threat actors have identified the MCP ecosystem as an attack surface worth investing in.

Simple detection works. 113 regex rules at 5.39 ms per skill scanned 96,096 skills in under 9 minutes. A single rule caught 52.7% of all detections. The technical barrier to pre-publication scanning is negligible. The barrier is organizational: registries must choose to deploy it.

Coordinated defense is emerging but incomplete. 1,302 threats in our scan, 1,184 in Antiy CERT's, and 554 entries in the Threat Cloud blacklist represent a growing shared intelligence resource. But standards for threat sharing, severity classification, and coordinated response across registries remain urgently needed.

All detection rules, scan infrastructure, and aggregate data are open source at .

Data Availability

The ATR rule set (113 rules, YAML) and detection engine are MIT-licensed. Aggregate scan statistics are published in the ATR repository. Individual threat reports are accessible through the Threat Cloud API with appropriate access controls. The PINT and SKILL.md benchmark datasets are available from their respective maintainers.

References

- [1] Anthropic, “Model Context Protocol specification,” , 2024.
- [2] Anthropic, “MCP specification: Authentication and transport,” , 2024–2026.
- [3] ATR Project, “Agent Threat Rules: Open detection standard for AI agent threats,” , 2026.
- [4] ATR Project, “Agent Threat Rules v2.0.0 release,” , April 2026.
- [5] ATR Project, “MCP ecosystem audit: 2,386 packages, 35,858 tool definitions,” ATR Documentation, March 2026.
- [6] ATR Project, “SKILL.md benchmark: 498 real-world skill descriptions,” ATR Documentation, April 2026.
- [7] Lakera, “PINT: Prompt Injection Test dataset,” , 2025.
- [8] OWASP, “Agentic AI Top 10 Threats,” , 2026.
- [9] OWASP, “Top 10 for Large Language Model Applications,” , 2025.
- [10] OpenSSF, “SAFE-MCP: Security Assessment Framework for MCP,” , 2026.
- [11] Cisco AI Defense, “Integrate ATR community rules as upstream detection (PR #79),” , March 2026.
- [12] MITRE, “CVE-2025-6514: Server-side request forgery in MCP server,” 2025.
- [13] MITRE, “CVE-2025-53773: Tool description injection enabling data exfiltration,” 2025.
- [14] MITRE, “CVE-2025-68143: Authentication bypass in MCP implementation,” 2025.
- [15] MITRE, “CVE-2025-68144: Authentication bypass chain (part 2),” 2025.
- [16] MITRE, “CVE-2025-68145: Authentication bypass chain (part 3),” 2025.
- [17] MITRE, “CVE-2025-49596: MCP tool invocation privilege escalation,” 2025.
- [18] MITRE, “CVE-2025-59536: MCP transport layer injection,” 2025.
- [19] Invariant Labs, “postmark-mcp: Email theft via tool description poisoning,” Security Advisory, 2025.
- [20] Socket Security, “SANDWORM_MODE: 19 typosquatted MCP packages,” , 2026.
- [21] Invariant Labs, “MCP security audit: 38% of servers implement zero authentication,” Technical report, 2025.
- [22] Antiy CERT, “ClawHub registry security analysis: 1,184 malicious packages identified,” Technical report, March 2026.
- [23] Checkmarx, “MCP security: Authentication gaps and supply chain attacks in the AI agent ecosystem,” Checkmarx Research, 2025–2026.
- [24] Snyk (Invariant Labs), “mcp-scan: Runtime monitoring for MCP connections,” , 2026.
- [25] Prompt Security, “MCP server risk scoring: 13,000+ servers assessed,” Technical report, 2026.

- [26] Docker, “MCP supply chain security: Gaps in package manager protections,” Docker Security Blog, 2026.
- [27] CyberArk, “Tool poisoning attacks on AI agents via MCP,” Technical report, 2025.
- [28] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, “Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection,” in *Proc. AISec Workshop*, 2023.
- [29] M. Ohm, H. Plate, A. Syber, and K. Peeters, “Backstabber’s knife collection: A review of open source software supply chain attacks,” in *Proc. DIMVA*, 2020.
- [30] P. Ladisa, H. Plate, M. Martinez, and O. Barais, “SoK: Taxonomy of attacks on open-source software supply chains,” in *Proc. IEEE S&P*, 2023.
- [31] Y. Liu et al., “Prompt injection attacks and defenses in LLM-integrated applications: A survey,” *arXiv:2310.12815*, 2023.
- [32] NousResearch, “Hermes Agent: Open-source AI agent framework,” , 2026.
- [33] NIST, “AI Agent Standards Initiative,” , 2026.