INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-11 14:21 UTC

AI Agent Skill Registries Are the New App Store Security Gap: 5% of 50K Skills Carry Multi-Stage Attack Chains

SECURITY ANALYSIS | HIGH | CVSS 7.5

SCC Item ID	SCC-STY-2026-0190
Type	Security Analysis
Severity	HIGH
CVSS Base Score	7.5
Affected Products	OpenClaw agent-skill registry; LLM agent platforms broadly (no specific vendor versions identified in source data)
Published	2026-06-11T10:00:24+00:00
Discovery Source	Rss:T1 Threatintel

Executive Summary

Unit 42 researchers audited nearly 50,000 skills in the OpenClaw agent-skill registry and found that roughly 1 in 20 contain multi-stage attack chains capable of credential theft, remote code execution, and agent hijacking. The registry operates with no automated integrity verification, meaning any enterprise running LLM agents in production may be silently executing adversary-controlled code at machine speed. This finding signals that AI agent extensibility has outpaced supply-chain security controls, creating an undefended attack surface analogous to the early mobile app store era before platform-level vetting existed.

Technical Analysis

Unit 42's audit of 49,943 skills in the OpenClaw agent-skill registry identified approximately 2,497 skills, roughly 5%, carrying multi-stage attack chains. The registry lacks automated integrity verification, meaning skills are published and consumed without cryptographic signing, provenance checks, or behavioral vetting. The attack surface mirrors early mobile app stores, where third-party extensibility was offered before supply-chain security controls existed at scale.

The reported attack capabilities map across multiple MITRE ATT&CK tactics. Credential exfiltration aligns with T1552 (Unsecured Credentials) and T1528, indicating skills may harvest API keys, tokens, or session credentials accessible to the agent runtime. Remote code execution paths align with T1059 (Command and Scripting Interpreter) and T1190 (Exploit Public-Facing Application), suggesting skills can invoke shell

interpreters or exploit agent-exposed interfaces. Agent hijacking techniques leverage T1574 (Hijack Execution Flow) and T1195 (Supply Chain Compromise), with the registry itself functioning as the compromised supply chain node. Obfuscation techniques (T1027, T1140) likely conceal malicious logic within skill definitions, complicating static review. Exfiltration over C2 channels (T1041) completes the kill chain once credentials or code execution are achieved.

The absence of integrity verification (CWE-494), combined with insufficiently protected credentials in the agent runtime environment (CWE-522) and the inclusion of functionality from an untrusted control sphere (CWE-829), creates compounding weakness. Enterprises deploying LLM agents in production typically grant those agents broad access to internal APIs, datastores, and workflow automation, meaning a malicious skill inherits that access surface immediately upon installation.

The threat actor profile remains unattributed in available source data; the pattern is consistent with opportunistic actors rather than tracked nation-state or criminal campaigns, suggesting low-friction publishing created a long tail of low-sophistication but high-volume malicious submissions. No specific attribution or campaign infrastructure has been identified in the source data.

Palo Alto Networks referenced Prisma AIRS in defensive context; NVIDIA has published a verified agent skills framework as a governance approach. The EU AI Act supply-chain provisions are also implicated given the cross-border distribution of registry skills.

Action Checklist

1. Step 1: Assess exposure, inventory every LLM agent deployment in your environment and identify which agent platforms consume skills, tools, or plugins from external registries, including OpenClaw or any registry without documented integrity verification.
2. Step 2: Audit installed skills immediately, pull the full list of skills installed across agent runtimes and cross-reference against your approved software inventory (CIS Controls v8 2.1 - Inventory and Control of Software Assets); treat any skill sourced from an unverified registry as untrusted until provenance is confirmed.
3. Step 3: Enforce integrity verification before execution, require cryptographic signing or hash verification for all agent skills before runtime execution; where the registry does not provide this, implement a compensating control at the platform or pipeline level; reference CWE-494 (Download of Code Without Integrity Check) as the root weakness to remediate.
4. Step 4: Apply least-privilege scoping to agent identities, audit the permissions granted to LLM agent runtime accounts and reduce them to the minimum required for documented workflows (NIST AC-6, Least Privilege); agent accounts with broad API or datastore access dramatically amplify the blast radius of a malicious skill.
5. Step 5: Enable credential monitoring for agent runtime environments, deploy monitoring on API keys, tokens, and session credentials accessible to agent processes; alert on credential access outside expected agent workflows; align to NIST AU-2 (Event Logging) and CIS Controls v8 8.2 (Collect Audit Logs).
6. Step 6: Update threat model, add 'malicious agent skill via unverified registry' as an explicit supply chain compromise vector (T1195) in your threat register; model the agent runtime as a privileged execution environment requiring the same scrutiny as a CI/CD pipeline.

7. Step 7: Communicate findings to leadership, brief the CISO and relevant product owners on the exposure conditions specific to your agent deployments; frame the risk as a supply-chain integrity gap, not a hypothetical, given the 5% malicious skill rate observed in source data.
8. Step 8: Monitor for follow-up disclosures, track Unit 42 research alerts (via Palo Alto security feeds), the OpenClaw registry changelog, and CISA AI Cyber Safety Institute advisories for published IOCs, updated audit findings, or registry remediation actions.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and legal counsel if any installed skill from the OpenClaw registry is confirmed to match a Unit 42-published malicious IOC, if anomalous credential access events are detected from agent runtime processes, or if agent identities are found to have accessed PII or regulated data datastores — each condition independently triggers potential breach notification obligations depending on applicable regulatory framework (GDPR, HIPAA, state breach laws).
Recovery Notes	Before restoring any agent runtime to production, verify that all skills have been replaced with provenance-confirmed, hash-verified alternatives sourced from registries with documented integrity controls, and that agent runtime accounts have been re-scoped to least-privilege permissions with new credentials issued to replace any that were accessible to unverified skills. Monitor agent process behavior and outbound network connections from agent hosts for a minimum of 30 days post-remediation, watching specifically for beaconing patterns or credential access anomalies consistent with a persisted malicious skill that survived initial cleanup. Retain all forensic artifacts collected during the incident for a minimum period consistent with your documented retention policy under NIST AU-11, as regulatory inquiries or insurance claims related to this supply-chain exposure may arise weeks after containment.
Forensic Artifacts	Agent runtime skill installation logs and package cache directories (e.g., /opt/agent/skills/, ~/.agent/plugins/, or platform-equivalent paths) — a malicious OpenClaw skill would leave behind its package files, dependency manifests, and install timestamps that establish when adversary-controlled code first entered the environment | Network capture or proxy logs of outbound connections from agent host systems to OpenClaw registry endpoints during skill download sessions — these contain the specific skill package URIs fetched, any redirect chains, and TLS certificate details that can be correlated against Unit 42 IOCs to confirm which malicious skills were retrieved | Cloud provider API audit logs (AWS CloudTrail, GCP Audit Logs, Azure Monitor) scoped to the agent runtime service account identity — a credential theft stage from a malicious skill executing T1528 would produce anomalous GetSecretValue, ListKeys, or equivalent calls from the agent process identity outside its documented workflow baseline | Process execution logs from agent host systems (Sysmon Event ID 4688 on Windows, auditd execve records on Linux) capturing child processes spawned by the agent runtime — a multi-stage attack chain delivering RCE via a malicious skill would manifest as unexpected interpreter invocations (bash, python, powershell) or network utilities (curl, wget, nc) spawned as children of the agent service process | Environment variable dumps from agent runtime processes (/proc//environ on Linux, captured at the time of investigation) — these reveal which API keys, tokens, and credentials were in scope for any malicious skill that executed during the exposure window, establishing the credential blast radius for impact assessment

Per-Action IR Details

Step 1: Assess exposure — inventory every LLM agent deployment in your environment and identify which agent platforms consume skills, tools, or plugins from external registries, including OpenClaw or any registry without documented integrity verification.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Establish IR capability and asset visibility before incidents occur

Controls: CIS 1.1 (IG1/IG2/IG3) — Establish and Maintain Detailed Enterprise Asset Inventory, CIS 2.1 (IG1/IG2/IG3) — Establish and Maintain a Software Inventory, NIST AC-20 — Use Of External Systems

Compensating: Run `find / -name 'agent_config*' -o -name 'skills.json' -o -name 'plugin_manifest*' 2>/dev/null` on agent host systems to locate runtime configuration files that enumerate registry endpoints. Cross-reference discovered registry URLs against a manually maintained allow-list in a shared spreadsheet. For containerized agents, `docker inspect | grep -i 'registry\|skill\|plugin'` to surface environment variables pointing to external skill sources.

Evidence: Before inventorying, capture agent runtime configuration files (skills.json, plugin_manifest, .env files) that record which registries are configured; capture network connection logs from agent host systems showing outbound connections to OpenClaw registry endpoints (api.openclaw.io or equivalent) to establish a baseline of what has already been fetched and executed.

Step 2: Audit installed skills immediately — pull the full list of skills installed across agent runtimes and cross-reference against your approved software inventory (CIS 2.1); treat any skill sourced from an unverified registry as untrusted until provenance is confirmed.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection & Analysis: Analyze collected data to determine scope and impact of adverse events

Controls: CIS 2.1 (IG1/IG2/IG3) — Establish and Maintain a Software Inventory, CIS 2.2 (IG1/IG2/IG3) — Ensure Authorized Software is Currently Supported, CIS 2.3 (IG1/IG2/IG3) — Address Unauthorized Software, NIST AU-6 — Audit Record Review, Analysis, And Reporting

Compensating: Extract installed skill metadata using the agent platform's CLI (e.g., `agentctl skills list --output json > installed_skills_$(date +%Y%m%d).json` or equivalent) and diff the output against your approved software inventory. For skills without a known-good hash, compute SHA-256 checksums with `sha256sum ` and compare against any available publisher-provided hashes. Use osquery (`SELECT * FROM file WHERE path LIKE '/opt/agent/skills/%'`) to enumerate skill files on Linux-based agent hosts.

Evidence: Capture the complete installed skills manifest from each agent runtime before making any changes; preserve skill package files and their metadata (install timestamps, source URLs, installer identity) from agent directories such as `/opt/agent/skills/`, `~/.agent/plugins/`, or equivalent platform-specific paths, as these will establish the chain of custody for any malicious skill identified later.

Step 3: Enforce integrity verification before execution — require cryptographic signing or hash verification for all agent skills before runtime execution; where the registry does not provide this, implement a compensating control at the platform or pipeline level; reference CWE-494 (Download of Code Without Integrity Check) as the root weakness to remediate.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment: Execute IR plan and mitigate further impact while preserving evidence

Controls: NIST SI-2 — Flaw Remediation, NIST CM-7 — Least Functionality, CIS 4.6 (IG1/IG2/IG3) — Securely Manage Enterprise Assets and Software, CIS 7.1 (IG1/IG2/IG3) — Establish and Maintain a Vulnerability Management Process

Compensating: Implement a pre-execution wrapper script that computes `sha256sum ` and compares against a locally maintained allow-list of approved hashes before the agent runtime loads any skill. Block execution and alert via syslog if the hash is absent or mismatched. For Python-based agent runtimes, intercept skill loading with a `importlib` hook that enforces hash validation. This can be maintained by a 2-person team as a plain-text hash manifest committed to a version-controlled repository.

Evidence: Before enforcing the block, capture network traffic (via tcpdump or Wireshark) of skill download sessions to document the registry endpoints, download URIs, and any redirect chains that delivered skills — this is critical to attributing which specific OpenClaw skills were fetched and to identifying whether a skill package was tampered with in transit.

Step 4: Apply least-privilege scoping to agent identities — audit the permissions granted to LLM agent runtime accounts and reduce them to the minimum required for documented workflows (NIST AC-6, Least Privilege); agent accounts with broad API or datastore access dramatically amplify the blast radius of a malicious skill.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment: Categorize and mitigate the incident while limiting further damage

Controls: NIST AC-6 — Least Privilege, NIST AC-2 — Account Management, NIST AC-3 — Access Enforcement, CIS 5.4 (IG1/IG2/IG3) — Restrict Administrator Privileges to Dedicated Administrator Accounts, CIS 6.1 (IG1/IG2/IG3) — Establish an Access Granting Process

Compensating: Run `grep -r 'AGENT_API_KEY\|AGENT_TOKEN\|AGENT_SECRET' /proc/*/environ 2>/dev/null` and `/proc/*/cmdline` to enumerate credentials currently in scope for running agent processes. On AWS/GCP/Azure, use the respective CLI (`aws iam simulate-principal-policy`, `gcloud iam roles describe`) to enumerate effective permissions granted to agent service accounts and identify over-privileged roles. Immediately rotate any credentials found to be broader than required and document the new minimal permission set.

Evidence: Before revoking permissions, capture a snapshot of current IAM role bindings, API key scopes, and datastore ACLs assigned to agent runtime accounts — this documents the blast radius that existed at time of exposure and is required for impact assessment; also capture process environment dumps (`/proc//environ`) from running agent processes to identify which credentials were accessible to potentially malicious skills at runtime.

Step 5: Enable credential monitoring for agent runtime environments — deploy monitoring on API keys, tokens, and session credentials accessible to agent processes; alert on credential access outside expected agent workflows; align to NIST AU-2 (Event Logging) and CIS 8.2 (Collect Audit Logs).

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection & Analysis: Monitor for indicators of adversary activity and correlate events across sources

Controls: NIST AU-2 — Event Logging, NIST AU-3 — Content Of Audit Records, NIST AU-6 — Audit Record Review, Analysis, And Reporting, NIST AU-12 — Audit Record Generation, CIS 8.2 (IG1/IG2/IG3) — Collect Audit Logs

Compensating: Deploy Sysmon with a configuration that includes ProcessAccess events (Event ID 10) targeting LSASS and credential store processes, and FileCreate events (Event ID 11) watching directories where agent runtimes cache tokens (e.g., `~/.agent/credentials`, `/tmp/agent_tokens/`). Write a Sigma rule detecting agent runtime process trees (e.g., python.exe or node.exe spawned under the agent service) that access credential file paths outside their normal working directory. Forward Sysmon logs to a centralized syslog server for retention per NIST AU-11 requirements.

Evidence: Preserve existing agent process logs and any credential vault access logs (e.g., AWS CloudTrail `GetSecretValue` events, HashiCorp Vault audit logs, Azure Key Vault diagnostic logs) from the period spanning skill installation to present — a malicious OpenClaw skill executing credential theft via T1528 (Steal Application Access Token) would produce anomalous API calls from the agent process identity that must be captured before log rotation overwrites them.

Step 6: Update threat model — add 'malicious agent skill via unverified registry' as an explicit supply chain compromise vector (T1195) in your threat register; model the agent runtime as a privileged execution environment requiring the same scrutiny as a CI/CD pipeline.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Update policies, improve detection, and share intelligence to prevent recurrence

Controls: NIST RA-3 — Risk Assessment, CIS 7.1 (IG1/IG2/IG3) — Establish and Maintain a Vulnerability Management Process, CIS 7.2 (IG1/IG2/IG3) — Establish and Maintain a Remediation Process

Compensating: Document the new threat vector in a plain-text or markdown threat register entry referencing MITRE ATT&CK T1195.001 (Compromise Software Dependencies and Development Tools) and T1059 (Command and Scripting Interpreter) as the execution mechanism once a malicious skill is loaded. Map the attack chain stages identified by Unit 42 (credential theft → RCE → agent hijacking) as sequential sub-techniques in your threat register so detection logic can be built per stage. No tooling cost required — this is a documentation and process update.

Evidence: Preserve the original Unit 42 research findings and any internal audit output from Steps 1–2 as supporting evidence for the threat model update; these documents establish that the threat is empirically validated (5% malicious rate in OpenClaw) rather than theoretical, which is required to justify the risk rating assigned in the threat register.

Step 7: Communicate findings to leadership — brief the CISO and relevant product owners on the exposure conditions specific to your agent deployments; frame the risk as a supply-chain integrity gap, not a hypothetical, given the 5% malicious skill rate observed in source data.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Communicate lessons learned and update organizational policies based on incident findings

Controls: NIST IR-7 — Incident Response Assistance, CIS 7.2 (IG1/IG2/IG3) — Establish and Maintain a Remediation Process

Compensating: Prepare a one-page briefing document that includes: (1) count of agent deployments consuming external registries from Step 1, (2) count of unverified skills from Step 2, (3) the Unit 42 finding of 5% malicious skill rate as the statistical basis for risk rating, and (4) the specific attack chain stages (credential theft, RCE, agent hijacking) with estimated business impact per stage. No tooling required — this is a structured communication artifact that a 2-person team can produce from the inventory outputs already collected.

Evidence: Attach the skills inventory output from Step 2, the IAM permission snapshots from Step 4, and the credential access log samples from Step 5 as appendices to the leadership brief — these concrete artifacts shift the framing from hypothetical risk to documented exposure and are necessary for securing remediation budget and executive support for registry governance controls.

Step 8: Monitor for follow-up disclosures — track Unit 42 and the OpenClaw registry for published IOCs, updated audit findings, or registry remediation actions; subscribe to CISA advisories for emerging AI agent supply-chain guidance.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Integrate threat intelligence and share information to improve future detection

Controls: NIST AU-13 — Monitoring For Information Disclosure, CIS 7.1 (IG1/IG2/IG3) — Establish and Maintain a Vulnerability Management Process

Compensating: Set up RSS or email subscriptions to the Unit 42 threat research blog (unit42.paloaltonetworks.com) and CISA advisories (cisa.gov/news-events/cybersecurity-advisories) filtered for keywords 'AI agent', 'LLM', 'skill registry', and 'OpenClaw'. Create a YARA rule targeting the SHA-256 hashes of any malicious skill packages published as IOCs by Unit 42 and deploy it to scan your agent skill directories on a daily cron job (`yara -r malicious_skills.yar /opt/agent/skills/`). When IOCs are published, immediately cross-reference against the installed skills manifest captured in Step 2.

Evidence: Maintain a dated archive of all external IOC feeds and Unit 42 disclosures received after initial exposure assessment, and log each cross-reference check against your installed skills manifest with a timestamp — this creates an auditable record demonstrating continuous monitoring and due diligence that is essential for any future regulatory inquiry or cyber insurance claim related to this supply-chain exposure.

Detection Guidance

Detection should focus on behavioral anomalies in agent runtime processes rather than static signatures, given the confirmed use of obfuscation (T1027, T1140) to conceal malicious skill logic.

Log sources to prioritize:

- Agent runtime execution logs: look for skills invoking shell interpreters (bash, cmd, PowerShell) or making outbound network connections not present in the skill's documented behavior (aligns to T1059, T1041).

- Credential store access logs: alert on any agent process accessing API key stores, token caches, or secret managers outside its expected workflow scope (T1552, T1528); align audit logging to NIST AU-2 and AU-12.

- Outbound network telemetry: flag agent processes initiating connections to external IPs or domains not in an approved allowlist, particularly over non-standard ports; this is the primary exfiltration-over-C2 signal (T1041).

- Supply chain integrity events: log every skill installation event with the source registry, version, and hash (if available); absence of a hash is itself a detection signal under CWE-494 logic.

Behavioral hunts to run now:

- Query for agent runtime processes spawning child processes (shell, interpreter, or network utility binaries) in the past 30 days.

- Hunt for credential access events originating from agent service accounts that are not associated with documented agent tasks.

- Review agent skill installation history for skills installed from the OpenClaw registry or any registry without a verified publisher tag.

D3FEND countermeasures to implement:

- D3-UAP (User Account Permissions): restrict agent runtime account permissions to documented minimum scope.

- D3-CRO (Credential Rotation): rotate any credentials accessible to agent runtimes immediately; prioritize API keys and service account tokens.

- D3-SFA (System File Analysis): monitor agent skill definition files and configuration stores for unauthorized modification post-installation.

- D3-LAM (Local Account Monitoring): monitor agent service accounts for access pattern deviations.

Note: No specific IOC hashes, domains, or IPs were present in the provided source data. Refer to the Unit 42 research publication for any indicators published alongside the audit findings.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`Pending — refer to Unit 42 research publication for published indicators`	Unit 42 audited 49,943 OpenClaw registry skills and identified ~2,497 containing multi-stage attack chains; any hashes, skill identifiers, or C2 infrastructure indicators are expected to be published in the Unit 42 report at https://unit42.paloaltonetworks.com/ai-agent-supply-chain-risks/ (human validation of URL required before operational use)	LOW

Framework Mappings

MITRE-ATTACK

- **T1552** — Unsecured Credentials
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1528** — Steal Application Access Token
- **T1059** — Command and Scripting Interpreter
- **T1574** — Hijack Execution Flow
- **T1140** — Deobfuscate/Decode Files or Information
- **T1027** — Obfuscated Files or Information
- **T1041** — Exfiltration Over C2 Channel
- **T1195** — Supply Chain Compromise
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan
- **SR-3** — Supply Chain Controls and Processes
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SI-2** — Flaw Remediation
- **CM-3** — Configuration Change Control
- **IA-5** — Authenticator Management

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

ISO-27001-2022

- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1552	Unsecured Credentials	Credential-Access
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1528	Steal Application Access Token	Credential-Access
T1059	Command and Scripting Interpreter	Execution
T1574	Hijack Execution Flow	Persistence
T1140	Deobfuscate/Decode Files or Information	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1041	Exfiltration Over C2 Channel	Exfiltration
T1195	Supply Chain Compromise	Initial-Access
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
Unit 42	https://unit42.paloaltonetworks.com/ai-agent-supply-chain-risks/	T3
	https://unit42.paloaltonetworks.com/ai-agent-supply-chain-risks/	T3
	https://developer.nvidia.com/blog/nvidia-verified-agent-skills-prov...	T3
	https://www.itsecurityguru.org/2026/05/19/securing-the-ai-supply-ch...	T3

Source	URL	Tier
Palo Alto Networks Secures the AI Agent Revolution with the Launch ...	https://www.paloaltonetworks.com/company/press/2025/palo-alto-netwo...	T3

Generated 2026-06-11 14:21 UTC by TJS Security Command Center