INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:32 UTC

Agentic AI Shell-Level Prompt Injection Enables Credential Exfiltration in Enterprise and Personal Deployments

SECURITY ANALYSIS | HIGH | CVSS 7.5

SCC Item ID	SCC-STY-2026-0015
Type	Security Analysis
Severity	HIGH
CVSS Base Score	7.5
Affected Products	OpenClaw (formerly Clawdbot, Moltbot) with shell access and messaging integrations; ClawdHub SKILLS ecosystem; integrations with WhatsApp, Telegram, Discord, Slack, Signal, iMessage
Published	2026-03-13
Discovery Source	Rss

Executive Summary

AI agents deployed with shell-level access and broad messaging integrations can be weaponized through prompt injection without a code-level exploit; the attack requires crafting an instruction in a delivery vector (e.g., spoofed email) that the agent will process, combined with knowledge of the agent's configuration and permissions. A proof-of-concept demonstrated full credential exfiltration from configuration files, and a scan of approximately 18,000 exposed OpenClaw instances suggests this is not a theoretical edge case. The risk is architectural: organizations deploying agentic AI without least-privilege controls have introduced a credential-theft surface that existing vulnerability management programs are not designed to detect.

Technical Analysis

The attack chain begins where most enterprise security programs are not looking: the instruction layer of an AI agent rather than its underlying code. OpenClaw (previously Clawdbot, Moltbot) operates with shell-level access, file system permissions, and integrations across WhatsApp, Telegram, Discord, Slack, Signal, and iMessage. An adversary who can place a crafted instruction into any content the agent processes, a spoofed email is the demonstrated vector, can direct the agent to read credential files such as clawdbot.json and exfiltrate API keys and gateway tokens over the agent's own communication channels. No CVE is assigned because no vulnerability in the conventional sense exists; the agent is functioning as designed. The weakness is

the absence of input validation, output controls, and least-privilege enforcement at the agent architecture level (CWE-272, CWE-77, CWE-522).

The attack maps cleanly to established MITRE ATT&CK techniques: T1566 (Phishing) as the initial injection delivery, T1059 (Command and Scripting Interpreter) for shell-level instruction execution, T1552 and T1552.001 (Unsecured Credentials / Credentials In Files) for token harvesting, and T1071 (Application Layer Protocol) for exfiltration over native agent communication channels. The ClawdHub SKILLS ecosystem adds a supply chain dimension (T1195), third-party plugins extend agent capabilities but also expand the trusted execution surface without commensurate security review.

A Reddit post in r/MachineLearning reports a scan of approximately 18,000 exposed OpenClaw instances; the methodology and accuracy of this scan are not independently verified, but the finding aligns with broader reports from Palo Alto Networks and Bitsight on widespread misconfiguration. Palo Alto Networks and Bitsight have both published analyses framing this as a signal of a broader architectural problem in agentic AI deployment rather than an isolated product flaw. JFrog's analysis specifically flags the credential storage pattern, secrets co-located with agent configuration files, as a systemic failure of secrets management discipline. Snyk's coverage reinforces that the supply chain risk through third-party SKILLS packages mirrors patterns seen in npm and PyPI ecosystem attacks.

The threat actor bar is low. Because no CVE or exploit is required, the attack class is accessible to adversaries with basic social engineering capability and knowledge of the target's agent configuration. However, the low technical barrier means this risk profile is accessible to both opportunistic actors and organized threat groups with social engineering resources; the absence of observed campaigns does not indicate absence of threat. Attribution to a named threat actor has not been established.

Action Checklist

1. Step 1: Assess exposure, inventory all agentic AI deployments (OpenClaw and equivalents) in your environment; identify which have shell-level access, file system permissions, or messaging integrations; flag any internet-facing or default-configured instances.
2. Step 2: Revoke and rotate credentials, audit clawdbot.json and equivalent agent configuration files for stored API keys, gateway tokens, or secrets; migrate credentials to a dedicated secrets manager (e.g., HashiCorp Vault, AWS Secrets Manager) and rotate any that were stored in plaintext.
3. Step 3: Apply least-privilege controls, restrict agent permissions to the minimum required for the defined task; disable shell access and file system permissions unless operationally justified; review and limit messaging integration scope.
4. Step 4: Audit third-party SKILLS and plugins, review installed ClawdHub SKILLS packages for provenance, update cadence, and permission scope; treat unvetted plugins as untrusted code with potential supply chain risk.
5. Step 5: Update your threat model, add prompt injection as an initial access and lateral movement vector in environments with agentic AI; map relevant TTPs (T1566, T1059, T1552, T1195) to existing detection and response playbooks.
6. Step 6: Brief leadership on the credential-theft surface introduced by agentic AI deployments, with emphasis on potential blast radius: compromised API keys or gateway tokens from a single agent could grant access to connected third-party services, customer data, or internal systems.

7. Step 7: Monitor for follow-up disclosures, track Palo Alto Networks, Bitsight, JFrog, and Snyk research channels for updated findings; watch for formal CVE assignment or vendor security advisories from OpenClaw maintainers.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate to external IR firm or forensic team immediately if: (1) any evidence of actual prompt injection attack is found (malicious messages in logs, unexpected credential exfiltration, agent process spawning shells), (2) rotated credentials show usage from unauthorized IP addresses or times post-rotation, or (3) more than 3 agents are found to have shell access plus plaintext credentials and internet-facing exposure.
Recovery Notes	After credential rotation and privilege restriction are complete: (1) run Step 1 inventory a second time to confirm all agents are now operating under least-privilege and no plaintext secrets remain in configs; (2) restore normal monitoring thresholds and baseline network/process activity from restricted agents to establish post-mitigation normal behavior; (3) schedule a 30-day post-incident review to assess detection rule effectiveness against the new threat model and document lessons learned.
Forensic Artifacts	Agent configuration files (clawdbot.json, .env, config.yaml, secrets.json) with modification timestamps and file hashes | Process execution logs (Linux: /var/log/auth.log, /var/log/syslog, auditd logs with rule on /opt/openai/*; Windows: Security Event Log 4688, PowerShell transcript logs if enabled) | Messaging platform audit logs from integrated services (Slack API audit log, Telegram message history, Discord audit log) covering 30 days prior to inventory date | Agent stderr/stdout logs and application logs showing message ingestion, command execution, and file I/O (e.g., /opt/openai/logs/*.log) | API access logs from integrated services (OpenAI API usage log, AWS CloudTrail for API key usage, Salesforce audit log, etc.) to detect unauthorized credential usage post-compromise or unusual API call patterns | Network connection logs: netstat snapshots, firewall logs, or Zeek/Suricata DNS/TLS logs showing outbound connections from agent processes to external command-and-control or exfiltration endpoints

Per-Action IR Details

Step 1: Assess exposure, inventory all agentic AI deployments (OpenClaw and equivalents) in your environment; identify which have shell-level access, file system permissions, or messaging integrations; flag any internet-facing or default-configured instances.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (preparation and prevention)

Controls: NIST IA-4 (identifier management), NIST CM-8 (system component inventory), CIS 4.1 (asset inventory and management)

Compensating: Execute `ps aux | grep -i claw` on all Linux/Unix hosts; use `wmic process list brief` on Windows to identify running AI agent processes. Query process command lines for shell flags (bash, /bin/sh). Scan configuration directories (/opt/openai, ~/.claw*, /etc/claw*) and parse config files for 'shell_enabled', 'file_access', 'messaging_integrations' keys using grep or jq. Document results in a CSV: hostname, process_name, shell_enabled, integrations, listening_ports, last_modified_date.

Evidence: Capture process listings (ps output, wmic output) and configuration file copies (clawdbot.json, equivalent configs) with timestamps before any inventory changes. Preserve file modification dates. Document network listeners (`netstat -tlnp` or `Get-NetTCPConnection` on Windows) to confirm internet-facing exposure.

Step 2: Revoke and rotate credentials, audit clawdbot.json and equivalent agent configuration files for stored API keys, gateway tokens, or secrets; migrate credentials to a dedicated secrets manager (e.g., HashiCorp Vault, AWS Secrets Manager) and rotate any that were stored in plaintext.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.2.4 (eradication) and §3.3 (post-incident)

Controls: NIST IA-5 (authentication and credential management), NIST IA-7 (cryptographic module management), NIST SC-7 (boundary protection), CIS 3.2 (address unauthorized software)

Compensating: Before rotation: use `grep -r 'api_key\|token\|secret\|password' clawdbot.json` to identify credential fields. Document plaintext credentials in a secured, isolated list (encrypted file, print to paper, destroy after rotation). Rotate credentials manually: (1) Generate new keys in each target service (OpenAI API, messaging platform gateways, etc.); (2) Update clawdbot.json with new values; (3) Restart agent process; (4) Log into each service and revoke old keys. For teams without Vault: store rotated secrets in OS-level credential managers (Windows Credential Manager via `cmdkey /add`, macOS Keychain via `security add-generic-password`, Linux pass or GnuPG-encrypted files in restricted directories with 0600 permissions).

Evidence: Capture plaintext credential files before any rotation (clawdbot.json, .env files, config dumps). Hash them (SHA-256) for chain-of-custody. Document which services used which credentials. Preserve API audit logs from each integrated service showing the old key's last use. Capture system timestamps and user IDs performing rotation for accountability.

Step 3: Apply least-privilege controls, restrict agent permissions to the minimum required for the defined task; disable shell access and file system permissions unless operationally justified; review and limit messaging integration scope.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.2.3 (containment) and §3.2.2 (analysis)

Controls: NIST AC-2 (account management), NIST AC-6 (least privilege), NIST AC-3 (access control), CIS 5.2 (ensure least privilege access)

Compensating: Modify clawdbot.json or equivalent: set 'shell_enabled': false, 'file_access_whitelist': ['/opt/openai/logs', '/tmp/agent-cache'] (only required directories), 'messaging_integrations': ['slack'] (remove unnecessary platforms like WhatsApp, Signal). For process-level restriction on Linux, wrap agent execution with `firejail --noproc --netfilter=~40 --read-only=/etc --read-write=/opt/openai/logs` to disable shell spawning and restrict network/file scope. Create a dedicated unprivileged user (useradd -r -s /bin/false claw-agent) and run agent under that user, removing sudo access. Document the business justification for any shell_enabled or messaging_integration exceptions in a change log.

Evidence: Capture original clawdbot.json before and after privilege restrictions (diff for audit trail). Document the denied capability list and business justification. Collect process capability snapshots (getpcaps on Linux) to verify unprivileged execution. Record messaging integration removals and affected dependent services so recovery is traceable.

Step 4: Audit third-party SKILLS and plugins, review installed ClawdHub SKILLS packages for provenance, update cadence, and permission scope; treat unvetted plugins as untrusted code with potential supply chain risk.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.1 (detection and analysis) and §3.2.2 (analysis)

Controls: NIST IA-4 (identifier management), NIST SC-7 (boundary protection), NIST SA-3 (system development life cycle), CIS 2.2 (ensure automatic security patches/updates)

Compensating: List installed SKILLS: `cat clawdbot.json | jq '.plugins[]' or grep -A 10 'SKILLS:' config.yaml`. For each plugin, document: (1) source URL/repository, (2) last commit date or version number, (3) author and organization, (4) required permissions (file read/write, network, shell). Cross-reference against public vulnerability databases (npm audit, GitHub Advisory Database, Snyk). For unvetted or old plugins (>6 months without update): create a removal plan and document business impact. Use `pip list` or equivalent to snapshot installed versions with timestamps for forensic

comparison if compromise is suspected.

Evidence: Capture installed SKILLS inventory with source URLs and version hashes before any removal. Screenshot GitHub/npm package pages showing last update dates. Preserve plugin code copies (git clone, pip download) for malware analysis if needed. Document plugin initialization logs showing load order and any errors during load.

Step 5: Update your threat model, add prompt injection as an initial access and lateral movement vector in environments with agentic AI; map relevant TTPs (T1566, T1059, T1552, T1195) to existing detection and response playbooks.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (preparation) and §3.1 (detection and analysis)

Controls: NIST SI-4 (information system monitoring), NIST IR-4 (incident handling), NIST RA-3 (risk assessment), CIS 4.2 (conduct risk assessment)

Compensating: Map MITRE ATT&CK to existing detections: (1) T1566.002 (phishing via messaging): create detection on incoming messages containing shell metacharacters or multi-line instruction patterns; look for '$(', '`', '|', '&&', '>>', 'cat', 'curl', 'wget' in messaging platform logs. (2) T1059 (command execution): monitor agent stdout/stderr and file system for unexpected file writes or external connections following message ingestion. (3) T1552 (unsecured credentials): alert on any regex matches for 'api_key=', 'token:', 'password' in agent process memory dumps or logs. (4) T1195 (supply chain): detect unexpected SKILLS/plugin downloads or version changes via package manager logs or config file hash changes. Create manual detections using syslog, `auditd` rules (`auditctl -w /opt/openai/config/ -p wa`), and messaging platform API webhooks to forward logs to a central alerting system.

Evidence: Document baseline threat model before changes (timestamp, author). Preserve existing detection rule versions. Capture messaging platform audit logs showing what messages the agent received during testing. Record timestamps of any test prompt injections for correlation with agent logs.

Step 6: Brief leadership on the credential-theft surface introduced by agentic AI deployments, with emphasis on potential blast radius: compromised API keys or gateway tokens from a single agent could grant access to connected third-party services, customer data, or internal systems.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §3.4 (post-incident activities) and §2.1 (preparation, stakeholder communication)

Controls: NIST RA-3 (risk assessment), NIST IR-2 (incident response organization), CIS 19.1 (ensure security responsibility)

Compensating: Prepare a risk summary document: (1) list all OpenClaw instances deployed and their integration scope (e.g., 'Agent A: Slack + AWS API + Salesforce API'); (2) estimate blast radius per agent (if API key compromised, which systems/customers could be accessed); (3) quantify: 'A single compromised agent API key grants access to ~150 enterprise customers' data via Salesforce'; (4) note remediation cost (cost of credential rotation, potential customer notification, audit/forensics). Use the inventory from Step 1 to build a dependency matrix. Present to CISO/VP Risk with action items and timelines. Frame as a 2-4 week operational security project, not a crisis.

Evidence: Preserve risk assessment document with timestamps and signoff. Document all briefing attendees and their acknowledgment. Retain the inventory matrix used to calculate blast radius for audit trail.

Step 7: Monitor for follow-up disclosures, track Palo Alto Networks, Bitsight, JFrog, and Snyk research channels for updated findings; watch for formal CVE assignment or vendor security advisories from OpenClaw maintainers.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §3.4 (post-incident activities) and §2.1 (preparation, vulnerability management)

Controls: NIST SI-5 (security alerts, advisories, and directives), NIST RA-5 (vulnerability scanning), CIS 3.10 (disable unused software)

Compensating: Set up RSS feed subscriptions or email alerts: Palo Alto Networks threat research (www.paloaltonetworks.com/research), JFrog security (jfrog.com/research), Snyk (snyk.io/blog/security), NIST CVE list (nvd.nist.gov). Use Google Alerts with keywords 'OpenClaw security', 'Clawdbot vulnerability', 'agentic AI prompt

injection', 'ClawdHub SKILLS exploit' to catch third-party reporting. Create a monthly checklist to review these sources and document findings in a shared log. If a CVE is assigned or an advisory released: immediately re-run Step 1 inventory and assess if your deployed version is affected; if yes, escalate to emergency change management.

Evidence: Maintain a log of all disclosures reviewed (date, source, URL, summary, impact on your environment). Document any new TTPs or attack variations discovered and map to your updated threat model from Step 5.

Detection Guidance

Detection for prompt injection in agentic AI environments requires instrumentation at the agent behavior layer, not just the network perimeter. Key signals to hunt for:

Credential file access: Monitor file system access logs for reads of agent configuration files (clawdbot.json, .env files, credential stores) by the agent process, especially when that access is not part of a documented workflow. Alert on any process reading credential files followed by outbound network activity.

Anomalous outbound communication: Review egress logs for agent processes initiating connections to external endpoints outside normal operational baselines, particularly over messaging platform APIs (WhatsApp Business API, Telegram Bot API, Discord webhooks, Slack webhooks). Exfiltration over these channels blends with legitimate agent traffic, baseline normal communication volume and destination sets.

Shell command execution: If shell access is enabled, log all commands executed by the agent process. Hunt for commands consistent with credential enumeration: cat, grep, find targeting configuration directories, or env variable dumps.

Unexpected instruction sources: If your agent logs inbound content it processes (emails, messages, documents), review for instructions that reference file paths, API tokens, or exfiltration destinations. These are not normal operational inputs.

SKILLS plugin behavior: Monitor for new plugin installations or permission escalation requests within the ClawdHub SKILLS ecosystem. Treat plugin updates from unverified publishers as a supply chain risk trigger.

Exposed instance detection: Run internal scans to identify any OpenClaw or equivalent agent instances accessible on non-internal network segments. Cross-reference with your asset inventory. Any internet-facing agent with default configuration should be treated as potentially compromised until reviewed.

Relevant log sources: EDR file access telemetry, network egress/proxy logs, agent-level audit logs (if enabled), secrets manager access logs, and email gateway logs for spoofed sender detection.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`clawdbot.json (file path pattern)`	Agent configuration file identified as primary credential storage target in demonstrated proof-of-concept; presence of API keys or gateway tokens in this file represents confirmed exposure pattern	HIGH

Type	Value	Context	Confidence
URL	`ClawdHub SKILLS registry (third-party plugin ecosystem)`	Supply chain risk surface; unvetted SKILLS packages may introduce malicious functionality into trusted agent execution context	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1195** — Supply Chain Compromise
- **T1190** — Exploit Public-Facing Application
- **T1195** — Supply Chain Compromise
- **T1566** — Phishing
- **T1552** — Unsecured Credentials
- **T1650** — Acquire Access
- **T1059** — Command and Scripting Interpreter
- **T1552** — Unsecured Credentials
- **T1059** — Command and Scripting Interpreter
- **T1071** — Application Layer Protocol
- **T1552.001** — Credentials In Files

NIST-800-53R5

- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-10** — Information Input Validation
- **IA-5** — Authenticator Management

OWASP-TOP10-2021

- **A03:2021** — Injection
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **16.10**
- **5.2**
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

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
- **CC9.2** — Manages risks associated with vendors and business partners

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program
- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195	Supply Chain Compromise	Initial-Access
T1190	Exploit Public-Facing Application	Initial-Access
T1566	Phishing	Initial-Access
T1552	Unsecured Credentials	Credential-Access
T1650	Acquire Access	Resource-Development
T1059	Command and Scripting Interpreter	Execution
T1071	Application Layer Protocol	Command-And-Control
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Security News	https://snyk.io/articles/clawdbot-ai-assistant/	T3
OpenClaw (formerly Moltbot, Clawdbot) May Signal the Next AI ...	https://www.paloaltonetworks.com/blog/network-security/why-moltbot-...	T3
[D] We scanned 18000 exposed OpenClaw instances and ... - Reddit	https://www.reddit.com/r/MachineLearning/comments/1r30nzv/d_we_scan...	T3
OpenClaw Security: Risks of Exposed AI Agents Explained | Bitsight	https://www.bitsight.com/blog/openclaw-ai-security-risks-exposed-in...	T3
Giving OpenClaw The Keys to Your Kingdom? Read This First - JFrog	https://jfrog.com/blog/giving-openclaw-the-keys-to-your-kingdom-rea...	T3

Generated 2026-03-29 18:32 UTC by TJS Security Command Center