INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:33 UTC

# OpenClaw AI Agent: Multiple Attack Vectors Enable Endpoint Compromise and Data Exfiltration

SECURITY ANALYSIS | HIGH | CVSS 7.5

| | |
|---|---|
| SCC Item ID | SCC-STY-2026-0014 |
| Type | Security Analysis |
| Severity | HIGH |
| CVSS Base Score | 7.5 |
| Affected Products | OpenClaw (formerly Clawdbot, Moltbot), ClawHub, Windows, macOS; integration surfaces include Telegram, Discord, GitHub |
| Published | 2026-03-14 |
| Discovery Source | Rss |

## Executive Summary

According to security reporting, China's CNCERT has issued an advisory against OpenClaw, an open-source autonomous AI agent, citing prompt injection vulnerabilities, malicious skill repositories, and exploitable default configurations that can lead to full endpoint compromise and data exfiltration. Researchers at PromptArmor demonstrated that indirect prompt injection via messaging app link previews can silently transmit sensitive data to attacker-controlled domains without any user interaction. Simultaneously, threat actors are distributing infostealer-laced fake OpenClaw installers through GitHub repositories. Huntress researchers documented that one malicious repository ranked as a top Bing search result for OpenClaw on Windows, indicating that standard user discovery paths led directly to the malicious installer.

## Technical Analysis

OpenClaw presents a multi-layered attack surface that extends well beyond the AI model itself. The most technically significant threat is indirect prompt injection (IDPI), also referred to as cross-domain prompt injection (XPIA). PromptArmor researchers demonstrated a specific exploitation path: an attacker embeds malicious instructions in web content that OpenClaw is asked to summarize or analyze. The agent interprets those instructions, constructs a URL pointing to an attacker-controlled domain, and appends sensitive in-context data as query parameters. When that URL renders as a link preview in Telegram or Discord, the data transmits automatically, no click required. This attack chain requires no direct access to the LLM and bypasses

conventional input validation controls focused on direct user input.

According to reporting on CNCERT's advisory, OpenClaw presents three additional risk categories beyond prompt injection. First, OpenClaw's privileged system access means a misinterpreted instruction can cause irreversible data deletion, an availability risk distinct from confidentiality concerns. Second, the ClawHub skill repository introduces a supply chain vector: malicious skills uploaded to the repository can execute arbitrary commands or deploy malware upon installation. Third, recently disclosed but unspecified security vulnerabilities in OpenClaw itself can be exploited to compromise the host system. The advisory explicitly calls out finance and energy sectors as high-consequence targets where successful exploitation could expose trade secrets, code repositories, or paralyze business operations. CNCERT's advisory does not assign specific CVE identifiers to the vulnerabilities cited; organizations should monitor NVD and OpenClaw's security channels for formal CVE assignments as they are published.

The fake installer campaign documented by Huntress represents a separate but concurrent threat that exploits OpenClaw's viral adoption. Attackers created malicious GitHub repositories mimicking legitimate OpenClaw installers, packaging infostealers including Atomic Stealer and Vidar Stealer alongside a Golang-based proxy malware called GhostSocks. The delivery method used ClickFix-style social engineering instructions, a technique that manipulates users into executing malicious commands under the guise of installation steps. Critically, one malicious repository ranked as a top Bing search result for OpenClaw on Windows, meaning users following normal discovery behavior were routed directly into the campaign. This campaign targeted all industries broadly, not a specific vertical, and covered both Windows and macOS environments.

These threat vectors converge around a common structural problem: open-source autonomous AI agents operate with elevated system privileges, consume untrusted external content as part of their core function, and are deployed into enterprise environments by users who may not be applying security-hardened configurations. OpenClaw's default management port exposure and plaintext credential storage compound the risk. Reporting indicates that the Chinese government has extended restrictions on OpenClaw to government agencies and military family members' personal devices, signaling that current mitigations are regarded as insufficient for sensitive environments. Security teams should treat this as a signal, not just news.

OpenClaw threat vectors span multiple attack surfaces (prompt injection, supply chain, endpoint compromise); traditional CVSS scoring applies to individual vulnerabilities, not compound threats. Organizations running OpenClaw should monitor CNCERT, NVD, and the OpenClaw project's own security advisories for CVE assignments and vulnerability disclosures.

## Action Checklist

1. Disable or restrict link preview rendering in any messaging platform integrated with OpenClaw; PromptArmor research confirms data exfiltration via IDPI occurs automatically upon the agent responding, with no user click required.
2. Block ClawHub skill installation from unverified publishers and disable automatic skill updates; treat third-party skills as untrusted code execution paths until a formal vetting process is in place.
3. Isolate OpenClaw in a container, block default management port exposure to the internet, and remove plaintext credentials from configuration files. These configurations are commonly exploited attack vectors for AI agents operating with elevated system privileges.
4. Audit GitHub repositories and internal documentation used to guide OpenClaw installation; Huntress documented a campaign that seeded malicious repositories ranking as top Bing search results, meaning standard discovery paths may be compromised.

5. Assign a monitoring task for CVE assignments against OpenClaw; CNCERT referenced exploitable vulnerabilities without publishing specific identifiers, and those disclosures may follow separately through NVD or the project's security channel.

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | URGENT |
| **Escalation Criteria** | Escalate to management immediately if: (1) OpenClaw is running in production with internet-facing management ports; (2) any plaintext API credentials, tokens, or database passwords are found in configuration; (3) malicious skill repositories or modified official repositories are detected in the installation audit; (4) outbound network connections to attacker-controlled domains (via PromptArmor IOC list if available from vendors) are confirmed. |
| **Recovery Notes** | Post-containment: (1) Rebuild OpenClaw instances from clean source (verified GitHub commit hash) with hardened configuration; (2) rotate all API keys, database credentials, and service account tokens that may have been exposed through plaintext config or indirect prompt injection; (3) audit downstream systems (databases, file shares, APIs) for unauthorized access patterns using NIST 800-61r3 §3.5 (post-incident activities) to confirm no lateral movement occurred. |
| **Forensic Artifacts** | Windows Event Log 4688 (process creation) + 3389 (RDP logon) filtered for OpenClaw process ancestry and timing; macOS `log show --predicate 'process == "clawdbot"'` for system logs \| OpenClaw configuration files (typically `~/.clawdbot/config.yaml`, `clawdbot.json`, environment variable exports) with full directory recursion and metadata timestamps \| Network traffic captures: `tcpdump -i any -w openclawai_traffic.pcap 'host '` and DNS resolution logs (`grep clawdbot /var/log/syslog` on Linux or macOS DNS query logs in `/var/log/system.log`) \| File system artifacts: ClawHub skill repository directory (`~/.clawdbot/skills/`), GitHub clones (`find ~ -name '.git' -path '*openclawai*'`), and browser download history for any OpenClaw-related installers \| Process memory dump if running: `sudo gcore ` (Linux/macOS) or Task Manager > Create dump file (Windows) to recover any in-memory API keys, credentials, or exfiltrated data fragments; web server access logs if OpenClaw exposed an HTTP interface |

### Per-Action IR Details

**Disable or restrict link preview rendering in any messaging platform integrated with OpenClaw; PromptArmor research confirms data exfiltration via IDPI occurs automatically upon the agent responding, with no user click required.**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2.1 (prevention and mitigation planning)

**Controls:** NIST 800-53 SI-4 (Information System Monitoring), CIS 6.1 (Establish and maintain detailed asset inventory)

**Compensating:** For Telegram: disable web preview in client settings (Settings > Privacy and Security > Link preview) and at bot level via BotFather API (`disable_web_page_preview=true` in sendMessage calls). For Discord: disable embed preview via role-based permissions (remove 'Embed Links' capability from bot role) and user settings (User Settings > Text & Images > Link Preview). For GitHub: disable Actions runner webhooks to external domains; audit integration tokens in Settings > Developer Settings > OAuth Apps and revoke unused integrations.

**Evidence:** Before disabling: capture current messaging app configuration exports, bot/integration settings files, and any existing link preview logs (Telegram: client.log; Discord: audit log exports via Guild Settings > Audit Log).

Document baseline outbound connections from OpenClaw process via `netstat -abno` (Windows) or `lsof -i -P -n` (macOS/Linux) to establish what domains are currently being contacted.

**Block ClawHub skill installation from unverified publishers and disable automatic skill updates; treat third-party skills as untrusted code execution paths until a formal vetting process is in place.**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2.1 (prevention controls) and §3.2.1 (detection engineering for malware)

**Controls:** NIST 800-53 SI-7 (Software, Firmware, and Information Integrity), NIST 800-53 AC-3 (Access Enforcement), CIS 2.4 (Disable Unnecessary Software), CIS 6.3 (Configure Data Access Control Lists)

**Compensating:** Implement a skill allowlist by modifying ClawHub configuration (typically `~/.clawdbot/config.yaml` or `clawdbot.json`) to set `skill_sources: [whitelist_only]` and enumerate trusted publishers by GPG fingerprint or SHA256 hash. Use `git log --all --oneline` and `git verify-commit` to validate skill repository commit signatures. For teams without code signing infrastructure: maintain a manual approval log (spreadsheet with skill name, version, publisher, date approved, approver name, hash of skill code) and script a pre-execution validation: `sha256sum skill_package.tar.gz | grep -f approved_hashes.txt || exit 1`.

**Evidence:** Before enforcement: snapshot current installed skills (`clawdbot list-skills --verbose` or equivalent; location varies by installation). Export ClawHub configuration files. Capture skill download history from `~/.clawdbot/skills/` or equivalent directory with file metadata (`ls -laR` with timestamps). Document any automatic update logs (usually in application logs or OS package manager logs). Extract all skill source URLs from configuration and document which are remote vs. local.

**Isolate OpenClaw in a container, block default management port exposure to the internet, and remove plaintext credentials from configuration files. These configurations are commonly exploited attack vectors for AI agents operating with elevated system privileges.**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2.1 (prevention and hardening) and §3.4.1 (containment strategy)

**Controls:** NIST 800-53 SC-7 (Boundary Protection), NIST 800-53 IA-2 (Authentication), CIS 5.1 (Establish and maintain firewall and router configuration standards), CIS 3.3 (Address Unauthorized Software)

**Compensating:** Use Docker with restricted network policy: create `Dockerfile` with `FROM python:3.9-slim`, run OpenClaw as non-root user, and bind only to localhost: `EXPOSE 8000` with `docker run --network none -p 127.0.0.1:8000:8000`. On macOS/Linux without Docker: use OS-native jailing (`chroot` + user restrictions or `systemd` PrivateTmp=yes). For credentials: store API keys in environment variables loaded from a separate secrets file (chmod 600, owned by service user only), never in config YAML. Use `grep -r "password\|api_key\|token" ~/.clawdbot/` to find plaintext secrets; replace with `${VARIABLE_NAME}` references. Restrict firewall with `ufw allow from 127.0.0.1 to any port 8000` (Linux) or `pfctl` rules (macOS).

**Evidence:** Before isolation: capture full process tree (`ps auxww | grep -i claw`), open file descriptors (`lsof -p `), and network connections (`netstat -abno` or `ss -tlnp`). Export complete configuration directory with hashes: `find ~/.clawdbot -type f -exec sha256sum {} \; > config_baseline.txt`. Document current firewall rules (`sudo iptables -L -n` or `ufw status numbered`). Snapshot current running container state if already containerized (`docker inspect `). Capture environment variables set for the process: `strings /proc//environ`.

**Audit GitHub repositories and internal documentation used to guide OpenClaw installation; Huntress documented a campaign that seeded malicious repositories ranking as top Bing search results, meaning standard discovery paths may be compromised.**

**NIST Phase:** Detection Analysis

**Reference:** NIST 800-61r3 §3.2.2 (malware analysis and supply chain investigation)

**Controls:** NIST 800-53 SI-4 (Information System Monitoring), NIST 800-53 RA-3 (Risk Assessment), CIS 2.1 (Establish and maintain a secure system hardening standard), CIS 6.2 (Ensure Software Is Obtained from Legitimate Sources)

**Compensating:** Audit GitHub repository metadata manually: (1) Check repository creation date, commit history timeline, and contributor identities via `git log --all --pretty=format:"%h %ai %an %ae %s"` on any cloned repo. (2) Compare official OpenClaw GitHub (verify domain ownership via GitHub's official security advisories or CNCERT link) against clones found in internal documentation. Use `curl -s https://api.github.com/repos/OWNER/openclawai | jq '.created_at, .updated_at, .stargazers_count'` to validate legitimacy. (3) Search internal wiki/documentation for OpenClaw install links: `grep -ri "github.*openclawai\|clawdbot" /path/to/internal/docs` and document all unique URLs. (4) For each URL, validate via domain registrar WHOIS (free tools: `whois domain.com`) and check GitHub Pages SSL certificate issuer. Malicious clones often use free/cheap certificates from bulk providers.

**Evidence:** Before auditing: capture all GitHub repo clones on the system: `find ~ -name '.git' -type d -exec git -C {} log --oneline -1 \;` with full path. Export browser history filtered for GitHub: `sqlite3 ~/.config/google-chrome/Default/History "SELECT url, title, last_visit_time FROM urls WHERE url LIKE '%github%openclawai%' ORDER BY last_visit_time DESC;"` (Chrome; adjust for Firefox/Safari). Document all DNS queries to GitHub-like domains: `grep github ~/.bash_history` or `sudo tcpdump -i any -w github_traffic.pcap 'host api.github.com'` (if available). Export Markdown/documentation files that reference OpenClaw installation: `find /opt/wiki -name '*.md' -exec grep -l openclawai {} \;`.

**Assign a monitoring task for CVE assignments against OpenClaw; CNCERT referenced exploitable vulnerabilities without publishing specific identifiers, and those disclosures may follow separately through NVD or the project's security channel.**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2.2 (tools and resources for incident handling) and §3.2.1 (detection)

**Controls:** NIST 800-53 SI-5 (Security Alerts, Advisories, and Directives), NIST 800-53 RA-5 (Vulnerability Scanning), CIS 4.9 (Address Unauthorized Software)

**Compensating:** Set up free monitoring using: (1) Subscribe to NVD RSS feed filtered for OpenClaw: `curl -s 'https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1-recent.json' | jq '.vulnerabilities[] | select(.cve.references[] | select(.url | contains("openclawai")))' `; trigger daily via cron. (2) Monitor GitHub security advisories for OpenClaw repository: add GitHub Releases RSS feed for the official repo (`https://github.com/OWNER/openclawai/releases.atom`) to email alert. (3) Subscribe to CNCERT/CC mailing list directly (China CNCERT publishes advisories at cert.org.cn; check for English RSS or email subscription). (4) Create a daily grep-based search of public vulnerability databases using `curl -s 'https://www.cvedetails.com/vulnerability-list/vendor_id-0/product_id-0/OpenClaw/' | grep -i 'cve-20'` (free, unstructured). (5) Set calendar reminder for manual check of OpenClaw official security page monthly.

**Evidence:** Document baseline: (1) current OpenClaw version installed: `pip show openclawai | grep Version` or `clawdbot --version`. (2) NVD CVE count for any OpenClaw product variant as of today: record count to detect new disclosures. (3) List any dependent packages: `pip freeze | grep -i clawai` or `npm list --depth=0`. Set up log rotation for monitoring output: designate a shared alerting file (`/var/log/openclawai_cve_alerts.log`) that all monitoring scripts append to, with read access for IR team.

## Detection Guidance

Monitor for: (1) Unexpected network connections from OpenClaw processes to unfamiliar domains, especially with query parameters containing file paths or data; (2) OpenClaw ClawHub skill installation from unverified sources; (3) Failed or suspicious OpenClaw installer executions and subsequent infostealer activity (Atomic Stealer, Vidar, GhostSocks); (4) Management port access logs for OpenClaw (default ports typically exposed in misconfigured deployments).

## Framework Mappings

**MITRE-ATTACK**

- **T1036.005** — Match Legitimate Resource Name or Location
- **T1566** — Phishing
- **T1041** — Exfiltration Over C2 Channel
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1203** — Exploitation for Client Execution
- **T1027** — Obfuscated Files or Information
- **T1552** — Unsecured Credentials
- **T1555** — Credentials from Password Stores
- **T1071.001** — Web Protocols
- **T1059** — Command and Scripting Interpreter
- **T1113** — Screen Capture
- **T1608.001** — Upload Malware

**NIST-800-53R5**

- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **SI-2** — Flaw Remediation
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-10** — Information Input Validation
- **AC-3** — Access Enforcement
- **SR-2** — Supply Chain Risk Management Plan

**OWASP-TOP10-2021**

- **A03:2021** — Injection
- **A01:2021** — Broken Access Control

**CIS-V8**

- **16.10**
- **6.1**
- **6.2**
- **15.1** — Establish and Maintain an Inventory of Service Providers

**SOC2-TSC**

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC7.4** — Responds to identified security incidents

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

**HIPAA-SECURITY**

- **164.312(a)(1)** — Access Control
- **164.308(a)(6)(ii)** — Response and Reporting

**ISO-27001-2022**

- **A.5.34** — Privacy and protection of personal information
- **A.5.21** — Managing information security in the ICT supply chain

**NIST-CSF-2**

- **GV.SC-01** — Cybersecurity supply chain risk management program

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1036.005 | Match Legitimate Resource Name or Location | Defense-Evasion |
| T1566 | Phishing | Initial-Access |
| T1041 | Exfiltration Over C2 Channel | Exfiltration |
| T1195.001 | Compromise Software Dependencies and Development Tools | Initial-Access |
| T1203 | Exploitation for Client Execution | Execution |
| T1027 | Obfuscated Files or Information | Defense-Evasion |
| T1552 | Unsecured Credentials | Credential-Access |
| T1555 | Credentials from Password Stores | Credential-Access |
| T1071.001 | Web Protocols | Command-And-Control |
| T1059 | Command and Scripting Interpreter | Execution |
| T1113 | Screen Capture | Collection |
| T1608.001 | Upload Malware | Resource-Development |

## Sources

| Source | URL | Tier |
|---|---|---|
| **Security News** | https://thehackernews.com/2026/03/openclaw-ai-agent-flaws-could-ena... | T3 |

| Source | URL | Tier |
|---|---|---|
| **[D] We scanned 18000 exposed OpenClaw instances and ... - Reddit** | https://www.reddit.com/r/MachineLearning/comments/1r30nzv/d_we_scan.. . | T3 |
| **Key OpenClaw risks, Clawdbot, Moltbot \| Kaspersky official blog** | https://www.kaspersky.com/blog/moltbot-enterprise-risk-management/5... | T3 |
| **Moltbot Gets Another New Name, OpenClaw, And Triggers Security ...** | https://www.forbes.com/sites/ronschmelzer/2026/01/30/moltbot-molts-... | T3 |
| **OpenClaw Explained: The Viral AI Agent Behind Moltbook** | https://letsdatascience.com/blog/openclaw-the-ai-agent-that-broke-t... | T3 |



Generated 2026-03-29 18:33 UTC by TJS Security Command Center