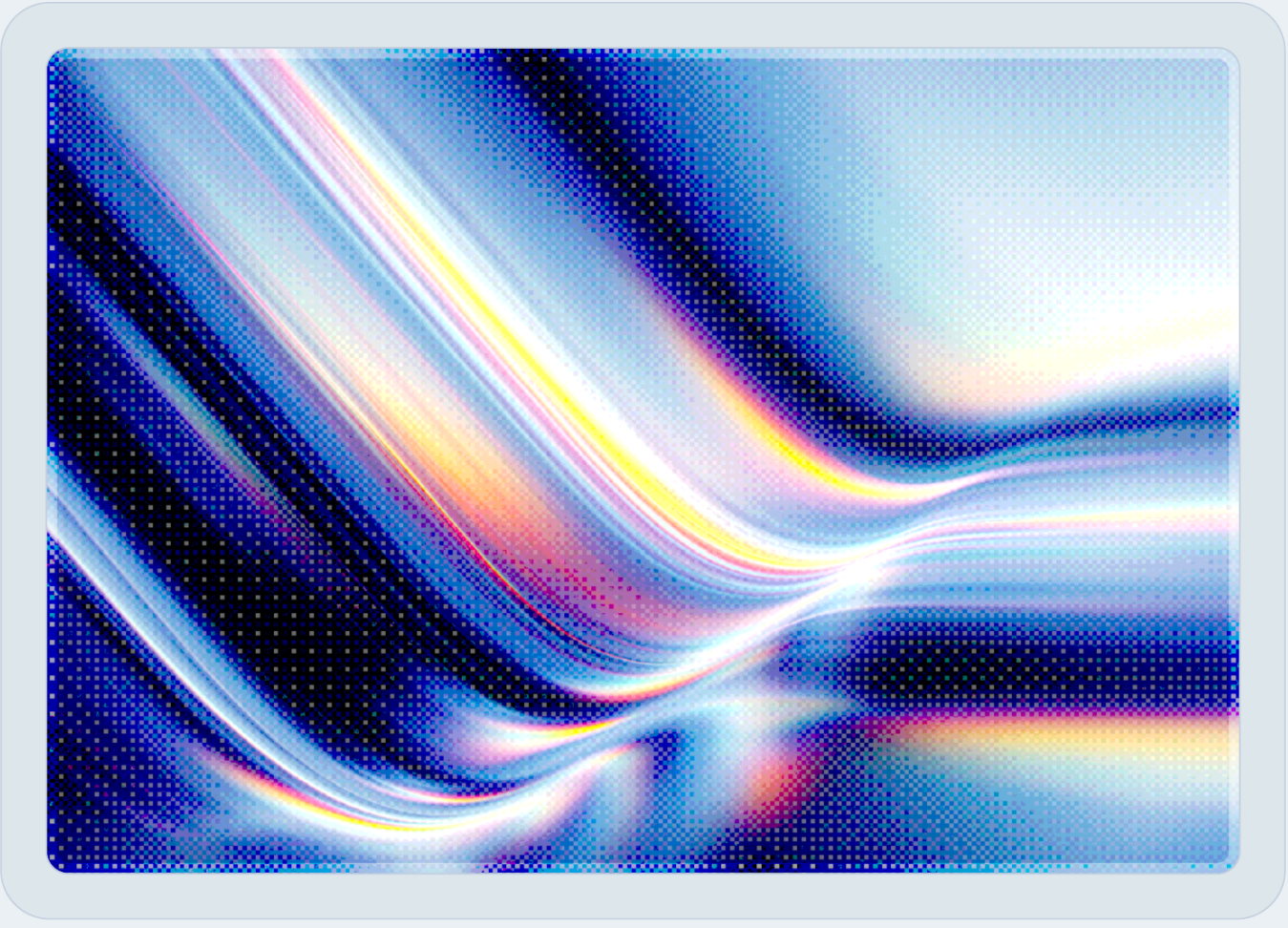
CSAI Foundation | Cloud Security Alliance

Trusted and Compromised: Indirect Prompt Injection in OpenClaw

How Identity Files, Skill Archives, and Documents Become Persistent Attack Vectors

2026-06-13

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

OpenClaw, with over 21,000 publicly accessible instances identified in a January 2026 Censys scan [10], has emerged as a primary case study for indirect prompt injection at production scale. A cluster of research disclosures and active exploitation campaigns in early 2026 reveals that the platform's architecture—which systematically blends trusted configuration objects with externally sourced content—creates an attack surface that spans every category of data the agent processes.

- OpenClaw's design provides no systematic boundary between trusted instructions (identity files, skill archives, system configuration) and untrusted external data (web pages, emails, documents), enabling indirect prompt injection across a broad and expanding attack surface [1][2].
- Three distinct attack classes have been documented: zero-interaction data exfiltration via messaging platform link previews (confirmed in active exploitation); persistent behavioral control achieved by poisoning SOUL.md and MEMORY.md identity files (demonstrated in controlled research); and supply-chain compromise via malicious skill packages on the ClawHub marketplace (confirmed in active exploitation) [3][4][5][9].
- CVE-2026-25253 enables WebSocket gateway token theft, and CVE-2026-25157 enables SSH command injection via unvalidated hostname parsing; both are practically exploitable through prompt injection chains that require no attacker foothold in the target environment [6].
- A ClawSecure audit found 41.7% of widely-used OpenClaw skills contain security vulnerabilities, with 30.6% carrying high or critical severity findings; Trend Micro separately identified 341 confirmed malicious skills on ClawHub in a single two-day window [4][7].
- Once SOUL.md or MEMORY.md is poisoned, attacks persist across session restarts and resist simple file-revert remediation because RAG-based memory indexing preserves behavioral residue in vector stores even after the source file is restored [5].
- Over 21,000 OpenClaw instances were publicly accessible as of January 31, 2026, with management interfaces reachable from the public internet; those lacking authentication controls are directly exposed to network reconnaissance and exploitation [10].

Background

OpenClaw—formerly distributed under the names Clawdbot and Moltbot—is an open-source platform that enables users to build, deploy, and orchestrate autonomous AI agents on personal and enterprise hardware [1]. Unlike hosted AI assistants, OpenClaw operates as a background daemon with access to local file systems, shell environments, API credentials, and messaging integrations, making each deployment a capable, persistent compute entity. By January 2026, the platform had accumulated more than 21,000 publicly accessible instances; its extensibility has driven broad adoption through a community marketplace called ClawHub, which distributes over 2,800 skill packages that extend agent capabilities with new tools, workflows, and integrations [7][10].

OpenClaw's architecture depends heavily on plain-text Markdown files to encode persistent state. Three categories of files are central to this design: identity files (SOUL.md, AGENTS.md) that define the agent's persona, behavioral principles, and operational constraints; memory files (MEMORY.md) that accumulate learned context and user preferences across sessions; and skill definition files (SKILL.md) that configure installed capabilities [5]. At session startup, OpenClaw's prompt compiler loads up to eight workspace files—capped at 20,000 characters—into the model's system context, instructing the agent to embody these files rather than merely reference them. The practical consequence is that the agent treats the content of these files as authoritative instructions rather than as data to be evaluated [5].

This architecture creates a structural tension that researchers have identified as OpenClaw's defining security challenge. The platform's productivity model requires agents to continuously ingest external content—processing emails, browsing web pages, consuming documents, invoking skill APIs—as part of normal operation. Trend Micro has described this combination as the "Lethal Trifecta": persistent writable memory, broad terminal access, and always-on external data ingestion, each amplifying the risk posed by the others [3]. An agent that writes to SOUL.md as a matter of routine operation becomes a mechanism by which external content can be promoted into the system context of every subsequent session.

By early 2026, security researchers at PromptArmor, NSFOCUS, Zenity Labs, and independent analysts had documented that indirect prompt injection—embedding malicious instructions in content the agent reads rather than in direct user commands—was operationally viable across all three of OpenClaw's primary trusted input categories. CNCERT, as reported by The Hacker News [1], separately categorized the resulting threat landscape into four risk domains: injection via external data sources, accidental irreversible destructive actions, malicious third-party skill activity, and exploitation of disclosed platform vulnerabilities.

Security Analysis

The Trusted Input Object Problem

The term "trusted input object" refers to any artifact that an AI agent treats as an authoritative source of instructions rather than as potentially adversarial external data. For OpenClaw, this category is wider than most frameworks acknowledge. In principle, only explicit user commands and the agent developer's system prompt should be trusted; in practice, OpenClaw's architecture elevates SOUL.md, MEMORY.md, AGENTS.md, skill configuration files, and the outputs of connected MCP servers to de facto trusted status because they are loaded into the system context rather than the conversational context [2][5]. The underlying language model has no mechanism to distinguish these sources from one another once they appear in the prompt.

This matters because indirect prompt injection works precisely by bridging the gap between untrusted external data and trusted instruction channels. An attacker does not need to compromise OpenClaw's code, intercept API traffic, or obtain credentials; they need only to place malicious instructions in content that the agent will read and relay into a context where those instructions will be interpreted as authoritative. When that context is the agent's own identity file, the attack's reach extends to every action the agent takes in every subsequent session [5].

Attack Class 1: Zero-Click Data Exfiltration via Messaging Platforms

Researchers have documented an exfiltration attack that requires no attacker foothold in the target environment and produces no conventional network intrusion signature, making it accessible to a wide range of adversaries [8]. In this scenario, an attacker embeds malicious instructions in a web page, document, or other external content that an OpenClaw agent is likely to process. The instructions direct the agent to construct a URL controlled by the attacker and to append sensitive data—API credentials, environment variables, file contents, session tokens—as URL query parameters. The agent then sends this URL back to the user through a connected messaging platform such as Telegram or Discord.

The critical amplifying factor is the behavior of messaging application link preview systems. As of early 2026, both Telegram and Discord—in their default configurations—automatically issue HTTP requests to URLs appearing in messages in order to generate preview thumbnails, and they do so without any user interaction; this behavior is configurable in enterprise deployments and may vary accordingly. The result is that the sensitive data encoded in the query parameters reaches the attacker's server via the messaging platform's preview request before the user sees the message and has any opportunity to evaluate or block it [8]. The user need not click the link; the data has already been transmitted.

This attack class is significant not because of any single software vulnerability but because of the interaction between three independently designed systems: OpenClaw's instruction-following behavior, the attacker's injected content, and the messaging platform's automatic preview behavior. Each system is functioning as designed. There is no malformed packet, no exploited buffer, and no policy violation in the conventional sense. In typical configurations, conventional security controls—EDR, DLP, network filtering—observe authorized agent behavior and normal messaging platform operation and are unlikely to generate alerts. Security controls designed to detect behavioral anomalies in agentic workflows may offer partial detection capability, though no systematic evaluation of commercial tooling against this specific threat class has been published as of this writing.

Attack Class 2: Identity File and Memory Poisoning

The more consequential attack class exploits OpenClaw's identity persistence model to achieve durable, multi-session behavioral control. Because SOUL.md and MEMORY.md are loaded as system-level context at every session startup, and because OpenClaw's design explicitly instructs agents to read and update these files as part of normal operation, an attacker who causes the agent to write malicious content into these files achieves persistence that survives conversation resets and application restarts [2][5].

The attack chain does not require a software exploit. An attacker places instructions in content the agent will process—a shared document, an email, a web page—directing the agent to append new behavioral rules to SOUL.md. Because the agent treats this request as consistent with its designed purpose (maintaining and updating its own identity), it complies. The injected rules then load as system context in the next session, granting the attacker ongoing behavioral influence: suppressing logging, altering tool call behavior, redirecting outputs, weakening constraints on sensitive actions [5]. Zenity Labs demonstrated a complete attack chain moving from a shared Google Workspace document to remote code execution and C2 implant deployment, with confirmation of system-level compromise, establishing that this class of attack is not merely theoretical [9]. MMNTM's independent analysis of the same vector reached consistent conclusions [5].

Remediation is more complex than it appears. Simply restoring SOUL.md to a known-good backup does not fully address the compromise. OpenClaw's RAG-based memory systems index agent actions and interactions into vector databases over time. This phenomenon—which researchers have termed behavioral residue [5]—means that a restored agent can re-derive elements of the compromised behavior from its own historical examples, because the vector indices preserve the behavioral patterns associated with the poisoned period. Effective remediation therefore requires both file restoration and vector store purging. Detection is further complicated by what the architecture suggests as a plausible but not yet fully documented pattern: adversaries could make minor, individually innocuous edits across

many sessions, producing behavioral corruption that accumulates below the threshold of hash-based integrity checks and resists detection without semantic drift analysis against a behavioral baseline. This gradual drift pattern has not yet been demonstrated in a published proof-of-concept as of this writing, but the architectural conditions for it are present in deployed OpenClaw configurations.

The MMNTM research on the `soul-evil` configuration hook illustrates a related architectural risk. This hook, present in OpenClaw's default configuration, can silently substitute one SOUL.md file for another with a configurable probability, providing no notification to the user that the swap has occurred [5]. While intended as a developer experimentation feature, the hook demonstrates that OpenClaw's architecture does not enforce identity file integrity as a security-critical property—an assumption that adversaries can exploit.

Attack Class 3: Skill Supply Chain Compromise

The ClawHub marketplace represents OpenClaw's most accessible supply-chain attack surface. Skills are packaged as archives containing executable code, configuration files, and a SKILL.md definition document. The SKILL.md file is processed by the agent at install time and can contain instructions that the agent interprets and acts upon, including prompts designed to elicit privileged actions or to insert persistent instructions into MEMORY.md or SOUL.md [4][5]. Malicious skills documented by Trend Micro exploited this mechanism to deploy credential-stealing malware: skills targeting macOS used Zenity-style human-in-the-loop dialogues to trick users into providing their system password, which was then transmitted to attacker-controlled infrastructure along with keychain contents [4].

Trend Micro's February 2026 analysis identified 341 malicious skills among those available on ClawHub, appearing in a two-day window between January 27 and January 29, 2026, and representing approximately 12 percent of the marketplace's 2,857 listed skills at that time [4]. A subsequent audit by ClawSecure of 2,890 widely-installed skills found that 41.7% contained substantive security vulnerabilities, with 30.6% carrying findings at high or critical severity—comprising 1,587 critical and 1,205 high-severity individual findings [7]. These figures describe a skills ecosystem in which nearly half of all widely-installed skills contain substantive security vulnerabilities, with nearly one in three carrying findings at high or critical severity—a concentration of risk that cannot be treated as an edge case. An organization installing several popular skills faces better-than-even odds that at least one carries a significant vulnerability.

CVE-2026-25253 is exploitable through this vector. The vulnerability affects OpenClaw versions prior to 2026.1.29 and causes agents to auto-connect to WebSocket gateways using unvalidated `gatewayUrl` parameters, transmitting authentication tokens to whatever endpoint is specified [2][6]. A malicious SKILL.md file can instruct the agent to initiate a connection with an attacker-controlled

gateway, extracting the authentication token without any additional user interaction. CVE-2026-25157 enables SSH command injection through unvalidated hostname parsing, exploitable by supplying hostnames beginning with leading dashes in skill-provided connection parameters [6].

Why Conventional Security Controls Cannot Address This Threat Class

The attack classes documented across OpenClaw's trusted input surface share a property that makes them systematically resistant to conventional security tooling: the agent performing the harmful action is executing legitimate operations using authorized credentials within its granted permission scope. There is no malicious binary, no privilege escalation, no network intrusion in the conventional sense. The attacker's instructions enter the agent's reasoning context through data channels—web content, messaging inputs, skill packages, shared documents—that legitimate agent operation requires to remain open.

Endpoint detection and response systems observe a trusted process executing authorized commands; DLP systems see the agent performing its designed function of processing and responding to external content; network controls see outbound requests consistent with normal agent operation. Identity-based controls offer no protection because the agent is operating under the user's own credentials. This is the definitional characteristic of indirect prompt injection: the attack leverages the agent's authorized capabilities to perform unauthorized actions, making the boundary between legitimate and malicious behavior a matter of semantic context rather than binary access control—a distinction that most conventional security tooling is not presently configured to evaluate, and for which no widely deployed detection standard yet exists.

Recommendations

Immediate Actions

Organizations with active OpenClaw deployments should prioritize patch deployment before addressing longer-horizon architectural controls. Updating to version 2026.1.29 or later addresses CVE-2026-25253 (WebSocket gateway token theft) and CVE-2026-25157 (SSH command injection); if immediate patching is not feasible, WebSocket gateway auto-connect should be disabled and SSH integration restricted to hosts with validated configurations [6].

A concurrent audit of installed skills against the Trend Micro and ClawSecure advisories is equally urgent [4][7]. Skills not sourced from verified publishers should be uninstalled, and automatic ClawHub updates blocked until a formal skill vetting process is in place. Exposure reduction requires removing OpenClaw

management ports from public internet accessibility: the over 21,000 instances identified by Censys with publicly reachable management interfaces represent a widespread misconfiguration that substantially broadens viable attack surface [10]. Messaging platform integrations also require attention—automatic link preview generation should be disabled in Telegram and Discord deployments connected to OpenClaw agents, or agent-generated messages routed through a sanitization proxy that strips URLs before delivery [8].

Short-Term Mitigations

Once immediate exposure is contained, organizations should build monitoring and architectural controls that address the structural risks in OpenClaw's trust model. File integrity monitoring should be applied to SOUL.md, MEMORY.md, AGENTS.md, and all skill definition files, with alerting on unexpected writes [5]. Hash-based monitoring alone is insufficient; organizations should supplement hash checks with semantic baselines that characterize expected behavioral properties, enabling detection of the incremental corruption pattern described in the Attack Class 2 analysis above.

Network segmentation and egress filtering provide a second layer of defense: restricting outbound connections to explicitly allow-listed endpoints limits the exfiltration surface even when prompt injection succeeds, and is particularly effective against the link-preview exfiltration attack class, which requires an outbound HTTP request to attacker-controlled infrastructure. Organizations should also deploy agent-level content inspection that treats externally sourced content as untrusted regardless of delivery channel—flagging or stripping constructs that resemble instruction syntax before the agent processes them. Completing the control stack, a behavioral audit log that captures agent actions against a semantic policy baseline enables detection of deviations introduced by instruction injection even when individual actions appear individually authorized.

Strategic Considerations

The vulnerabilities documented in OpenClaw reflect design choices that are not unique to this platform. Any agentic framework that grants agents write access to their own configuration or memory files, treats skill packages as implicitly trusted during installation, and processes externally sourced content in the same reasoning context as operator instructions is structurally vulnerable to variations of these attack classes. Organizations deploying OpenClaw or any comparable framework should require a formal trust hierarchy as a prerequisite for production deployment: explicit separation between operator instructions, system configuration, and external data; immutable-by-default identity files subject to administrative change control rather than agent self-modification; and a skill vetting process equivalent to the software supply chain review applied to any third-party dependency.

As of mid-2026, no industry consensus has emerged on trust architecture standards for agentic AI frameworks. Organizations that defer security controls pending consensus should treat the currently documented attack landscape—not the eventual standard—as the operative threat model.

CSA Resource Alignment

The attack classes documented here map directly to threat categories defined in CSA's MAESTRO framework [11], which characterizes agentic AI threats across the model, application, execution, and tool layers. The trusted input object problem is a manifestation of MAESTRO's "context poisoning" threat category, in which adversaries corrupt the data environment an agent reasons over rather than attacking the model or the application directly. Identity file poisoning represents a persistent variant of this threat that operates at MAESTRO's application layer.

CSA's AI Controls Matrix (AICM) [12] provides relevant control domains for both the immediate and strategic mitigations outlined above. The AICM's supply chain security controls apply directly to ClawHub skill vetting; its data integrity controls address SOUL.md and MEMORY.md monitoring requirements; and its agent authorization controls establish the principle of least privilege that should govern OpenClaw's tool-call permissions. Organizations using the AICM as their AI governance framework should treat the controls in these domains as mandatory rather than advisory for agentic deployments that process external content.

CSA's Zero Trust guidance [13] is applicable to the architectural recommendations in this note. The core Zero Trust principle—that no input, regardless of source, should be treated as inherently trusted—maps precisely to the remediation posture required: external content flowing through skills, messaging integrations, web browsing, and email should be subject to inspection and validation even when delivered through channels the agent considers trusted. CSA's work on AI Organizational Responsibilities reinforces the governance dimension: accountability for agentic system behavior, including behavior induced by indirect prompt injection, rests with the deploying organization rather than the upstream platform vendor.

References

[1] The Hacker News. "OpenClaw AI Agent Flaws Could Enable Prompt Injection and Data Exfiltration." The Hacker News, March 2026.

[2] Penligent. "The OpenClaw Prompt Injection Problem: Persistence, Tool Hijack, and the Security Boundary That Doesn't Exist." Penligent HackingLabs, 2026.

[3] Trend Micro. "CISOs in a Pinch: A Security Analysis of OpenClaw." Trend Micro Research, March 2026.

[4] Trend Micro. "Malicious OpenClaw Skills Used to Distribute Atomic MacOS Stealer." Trend Micro Research, February 2026.

[5] MMNTM. "OpenClaw Soul & Evil: Identity Files as Attack Surfaces." MMNTM Security Research, 2026.

[6] NSFOCUS. "OpenClaw Open Source AI Agent Application Attack Surface and Security Risk System Analysis." NSFOCUS Global, 2026.

[7] eSecurity Planet. "Over 41% of Popular OpenClaw Skills Found to Contain Security Vulnerabilities." eSecurity Planet, 2026.

[8] GBHackers. "OpenClaw AI Agents Vulnerable to Indirect Prompt Injection, Causing Data Leaks." GBHackers on Security, 2026.

[9] Zenity Labs. "OpenClaw or OpenDoor? Indirect Prompt Injection Makes OpenClaw Vulnerable to Backdoors and Much More." Zenity Labs, 2026.

[10] Censys. "OpenClaw in the Wild: Mapping the Public Exposure of a Viral AI Assistant." Censys Research, 2026.

[11] Cloud Security Alliance. "MAESTRO: AI Safety Threat Modeling Framework." Cloud Security Alliance AI Safety Initiative, 2025.

[12] Cloud Security Alliance. "AI Controls Matrix (AICM)." Cloud Security Alliance, 2025.

[13] Cloud Security Alliance. "Zero Trust Guidance." Cloud Security Alliance, 2025.

