CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Securing OpenClaw in the Enterprise: A Zero Trust Approach to Agentic AI Hardening

Enterprise Adoption Framework, Zero Trust Architecture, and Operational Hardening Guide

Unofficial AI-assisted Research

2026-03-31

© 2026 Cloud Security Alliance. Unofficial AI-assisted Research.

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Table of Contents

© 2026 Cloud Security Alliance. Unofficial AI-assisted Research.

Executive Summary

OpenClaw has become the dominant open-source agentic AI platform, accumulating over 250,000 GitHub stars and achieving 22 percent shadow-IT penetration among surveyed enterprises within months of its January 2026 launch. Its rapid adoption has far outpaced its security maturity, producing a concentrated disclosure sequence that includes one-click remote code execution vulnerabilities, Docker sandbox escapes, the largest confirmed supply chain attack against AI agent infrastructure, and persistent behavioral manipulation through the agent's core identity file. As of March 2026, the platform has accumulated 169 security advisories — five rated Critical, 58 High, 96 Medium, and 10 Low — while a separate Kaspersky audit identified 512 vulnerabilities, eight classified as critical. The ClawHavoc supply chain campaign planted between 341 and 1,467 malicious skills on ClawHub, combining prompt injection with the Atomic macOS Stealer to harvest credentials across thousands of deployments. SecurityScorecard's STRIKE team identified over 135,000 OpenClaw instances exposed to the public internet across 82 countries.

This paper consolidates the Cloud Security Alliance's complete body of research on OpenClaw security into a single authoritative reference. Part I documents the full security landscape: the vulnerability disclosure sequence, the ClawHavoc campaign, internet exposure at scale, the SOUL.md persistence attack surface, and the MCP security crisis. Part II provides an enterprise adoption framework that begins with a decision matrix for platform selection, surveys the agentic standards landscape, and defines a four-phase maturity model from shadow IT containment through production hardening. Part III establishes why Zero Trust is the necessary architectural philosophy for governing agentic AI, maps CSA's 11 Zero Trust Guiding Principles and five-step methodology to agent governance, and addresses agent identity, network security, MCP tool invocations, agent-to-agent delegation, and autonomy governance. Part IV delivers operational hardening controls across installation security, skill vetting, runtime sandboxing, file system and credential management, and behavioral monitoring with incident response playbooks. Part V aligns the complete control set to OWASP Agentic Top 10, MITRE ATLAS, MAESTRO, and AICM frameworks, maps regulatory compliance obligations, and provides resource planning guidance. The paper concludes with an architect quick guide and prioritized hardening checklist that organizations can begin implementing immediately.

The central thesis is straightforward: enterprises must build a governance architecture — an agentic control plane — that separates security policy from agent execution, providing identity governance, authorization enforcement, data control, and behavioral observability that scale alongside agent capabilities regardless of which execution platform underlies them.

Part I: The OpenClaw Security Landscape

1. Platform Overview and the Security Disclosure Sequence

OpenClaw emerged in November 2025 as "Clawdbot," a personal AI agent built by Austrian developer Peter Steinberger, founder of PSPDFKit. Two trademark-driven renames later — to Moltbot on January 27, 2026, then to OpenClaw three days later — the project reached 250,000 GitHub stars and 47,700 forks by early March 2026.

© 2026 Cloud Security Alliance. Unofficial AI-assisted Research.

Steinberger announced on February 14, 2026 that he was joining OpenAI, and that OpenClaw would transition to an independent open-source foundation with OpenAI's support [1][2]. The platform's extensibility is its defining characteristic and its primary security liability. OpenClaw operates through a skill-based architecture in which community-contributed modules extend the agent's capabilities to interact with email, calendars, filesystems, databases, APIs, and messaging platforms. The agent maintains persistent memory, communicates through WebSocket-based local gateways, and can be connected to the Moltbook social network where agents interact autonomously on a four-hour heartbeat cycle.

Palo Alto Networks described this combination as the "lethal quartet" — an agent with simultaneous access to private data, exposure to untrusted content, the ability to communicate externally, and persistent memory that enables time-shifted attacks [3][4]. Each characteristic is manageable in isolation; their combination in a single entity creates a risk configuration where a successful prompt injection in any content the agent processes can leverage all four characteristics to exfiltrate data, establish persistence, communicate with attacker infrastructure, and maintain adversarial influence across future sessions.

The security disclosure sequence began with CVE-2026-25253 (CVSS 8.8), disclosed publicly on February 3, 2026 after a silent patch in v2026.1.29. This vulnerability allowed any website to craft a link that would cause a visiting user's OpenClaw instance to transmit its authentication credentials to an attacker-controlled server, enabling full remote code execution in milliseconds — SOCRadar characterized it as a "1-Click RCE Kill Chain" with public exploit code available on GitHub within days of disclosure [5][6]. The second wave arrived on February 13, 2026 with GHSA-g27f-9qjv-22pm, a log poisoning vulnerability in which OpenClaw's gateway wrote WebSocket request headers verbatim to a log endpoint accessible on TCP port 18789. Because OpenClaw agents periodically review their own logs, an unauthenticated attacker could inject crafted LLM instructions via WebSocket headers that the agent would subsequently process as directives — an indirect prompt injection path requiring no authentication whatsoever [7].

The third disclosure came on February 25, 2026, when Oasis Security published the ClawJacked vulnerability chain. Three compounding design failures — absent WebSocket origin enforcement, disabled rate limiting for loopback connections, and automatic device pairing from localhost — allowed any malicious website to silently seize full authenticated control of a locally running OpenClaw instance with no user interaction beyond visiting a page [8][9]. CVE-2026-24763 (CVSS 8.8) documented a Docker sandbox escape via PATH manipulation that negated the isolation controls many organizations assumed were protecting them [10]. Additional critical CVEs included CVE-2026-25157 (SSH command injection on macOS, CVSS 7.8), CVE-2026-26322 (SSRF in Gateway tool, CVSS 7.6), CVE-2026-27001 (prompt injection via unsanitized workspace path, CVSS 8.6), CVE-2026-22172 (scope elevation in WebSocket shared-auth, CVSS 9.4), CVE-2026-30741 (RCE via request-side prompt injection), and CVE-2026-32913 (fetch-guard credential leakage on redirect, CVSS 8.8).

The following table summarizes the most operationally significant vulnerabilities and incidents from OpenClaw's first three months.

Date	Identifier	Description	CVSS	Status
Jan 24, 2026	CVE-2026-25253	One-click RCE via WebSocket token theft	8.8	Patched v0.5.0

Date	Identifier	Description	CVSS	Status
Jan 27, 2026	ClawHavoc	First malicious skills uploaded to ClawHub	–	Ongoing
Jan 31, 2026	ClawHavoc surge	335 coordinated malicious skill uploads	–	Partial removal
Feb 2026	CVE-2026-24763	Docker sandbox escape via PATH manipulation	8.8	Patched v2026.1.29
Feb 2026	CVE-2026-25157	SSH command injection (macOS)	7.8	Patched v2026.1.29
Feb 2026	CVE-2026-26322	SSRF in Gateway tool	7.6	Patched v2026.2.14
Feb 2026	CVE-2026-27001	Prompt injection via unsanitized workspace path	8.6	Patched v2026.2.15
Feb 2026	CVE-2026-27002	Container escape via bind mount config injection	Moderate	Patched v2026.2.15
Feb 2026	GHSA-M8V2	Sandbox escape via symlink manipulation	8.2	Patched v2026.2.23+
Feb 2026	ClawJacked	Localhost WebSocket brute-force / auto-pairing	High	Patched v2026.2.25
Feb 2026	CVE-2026-22172	Scope elevation in WebSocket shared-auth	9.4	Patched
Mar 2026	CVE-2026-30741	RCE via request-side prompt injection	High	See advisory
Mar 2026	CVE-2026-32913	fetch-guard credential leakage on redirect	8.8	See advisory

The minimum safe version for enterprise deployment is v2026.2.26 or later, which addresses all known critical and high-severity CVEs through early March 2026.

2. Supply Chain Compromise: The ClawHavoc Campaign

The ClawHub skill registry — OpenClaw's primary distribution channel for community extensions — harbored the largest confirmed supply chain attack against AI agent infrastructure. An initial audit by Koi Security identified 341 malicious skills; subsequent analysis by Snyk's ToxicSkills initiative evaluated 3,984 skills and found 1,467 malicious entries, with 91 percent combining prompt injection with traditional malware payloads [11][12]. The campaign distributed the Atomic macOS Stealer through fake prerequisite installation prompts, malicious MSI installers hosted on GitHub, and hidden MCP server endpoints tunneled through bore.pub. Attack vectors included 127 skills that directly requested private keys and passwords, and base64-encoded payloads designed to evade casual code review. Cisco's analysis found nine vulnerabilities in the single most-downloaded ClawHub skill, indicating that even popular, widely-used skills may contain critical security flaws [13]. Critically, 36 percent of all ClawHub skills were found to contain detectable prompt injection — not malware in the traditional sense, but instructions embedded in skill descriptors that redirect agent behavior in ways the user does not intend or observe [12].

The root cause was the absence of meaningful gatekeeping in the ClawHub marketplace. At the time of the campaign, the only requirement for publishing a skill was a GitHub account at least one week old. There was no automated static analysis, no code review, no cryptographic signing, no behavioral sandboxing, and no reputation system to distinguish established publishers from anonymous accounts uploading malware. The estimated total of 800+ malicious skills across 4,000 total entries represented approximately 20 percent of the entire registry [11].

The companion Moltbook platform compounded the supply chain exposure. Wiz Research discovered a database misconfiguration that left 1.5 million API authentication tokens, 35,000 email addresses, and private inter-agent messages accessible to any authenticated user. Combined with Moltbook's heartbeat mechanism — in which connected agents fetch and execute new instructions from moltbook.com every four hours — the exposed tokens represented potential command-and-control access to a substantial fraction of the 1.5 million agents registered on the platform [14][15]. Meta acquired Moltbook on March 10, 2026, which may stabilize its security posture, but the architectural pattern of agents polling a centralized service for instructions at regular intervals remains an inherent risk vector regardless of ownership [16].

3. Internet Exposure and Shadow IT

The scale of unmanaged OpenClaw deployment is substantial. Internet-wide scanning by SecurityScorecard's STRIKE team identified over 135,000 exposed instances across 82 countries, with 42,665 publicly accessible control panels and 15,200 instances vulnerable to the CVE-2026-25253 RCE chain. Over 53,000 instances correlated with prior breach activity [17]. JFrog research reported that 93.4 percent of publicly reachable instances in their sample exhibited authentication bypass vulnerabilities [18]. Token Security reported that approximately 22 percent of employees among its enterprise customer base were running OpenClaw, overwhelmingly without IT knowledge or approval [19]. IBM's 2025 Cost of a Data Breach Report found that shadow AI was present in 20 percent of breaches and added $670,000 to average costs — a figure that predates the OpenClaw-specific supply chain and credential exposure incidents [20].

The root cause of internet exposure was architectural: OpenClaw binds its gateway to 0.0.0.0:18789 by default, listening on all network interfaces rather than restricting to localhost. For software that holds persistent filesystem permissions and can execute arbitrary agent-driven actions on the host machine, this default represents a fundamental misalignment between the software's capability surface and its network exposure posture. Additionally, OpenClaw

broadcasts its presence via mDNS/Bonjour by default, leaking filesystem paths and SSH port metadata to any device on the local network. These numbers establish that enterprise OpenClaw adoption is not a future planning exercise — it is a present-tense shadow IT reality that security teams must address with pragmatic governance rather than prohibition.

4. The SOUL.md Persistence Attack Surface

OpenClaw maintains a persistent context file called SOUL.md that defines the agent's identity, behavioral boundaries, personality, and operational constraints. This file is injected into every interaction as part of the system prompt, making it the foundational document that governs how the agent interprets instructions and constrains its own behavior. From a security perspective, SOUL.md is both a configuration file and a policy enforcement mechanism — and because it is stored as a plaintext markdown file on the local file system, it is accessible to any process or prompt injection attack that can write to the directory where OpenClaw stores its state. Modification of SOUL.md does not trigger any integrity verification or alert; changes take effect immediately and persist across all subsequent sessions [21].

Penligent AI demonstrated several attack patterns exploiting this surface. The "Ship of Theseus" attack uses gradual incremental edits over days: Day 1 adds "efficiency language," Day 10 reinterprets efficiency as "skipping confirmation," Day 30 interprets it as "executing commands without review" — with each individual change appearing legitimate. Soul Packs, downloadable SOUL.md templates, may contain steganographic injections: base64-encoded instructions, zero-width Unicode characters, or commented sections invisible to human review but processed by the model [22]. HiddenLayer demonstrated that modifying the companion HEARTBEAT.md file can install persistent instructions that cause the agent to silently poll an attacker-controlled server for commands, converting the agent into a durable command-and-control implant that survives reinstalls and cloud sync [23]. Additionally, OpenClaw's configuration restrictions mention `config.apply` and `config.update` but omit `config.patch`, creating a gap that allows unrestricted configuration modification [21].

5. The MCP Security Crisis

The Model Context Protocol has emerged as one of the most significant attack surfaces in the agentic AI ecosystem. Between January and February 2026, security researchers filed over 30 CVEs targeting MCP servers, clients, and infrastructure [24]. A BlueRock Security analysis of 7,000 MCP servers found that 43 percent contained command injection vulnerabilities, 38 percent lacked authentication entirely, 82 percent of file operations were vulnerable to path traversal, and 67 percent had code injection risk [24][25]. The root causes are structural: missing input validation, absent authentication, blind trust in tool descriptions, and a protocol design that prioritizes developer ergonomics over security.

Three MCP-specific attack patterns are particularly relevant to OpenClaw deployments. Tool poisoning embeds adversarial instructions in tool description metadata — text that the agent reads to understand tool capabilities but that human operators typically do not review in full. A poisoned tool description can redirect the agent to exfiltrate sensitive files, transmit credentials to attacker-controlled endpoints, or modify its own configuration while presenting benign behavior to the user. Rug pull attacks exploit MCP's ability to modify tool definitions between sessions: a server presents benign capabilities during initial approval, then silently switches to malicious definitions in subsequent sessions without triggering re-authorization. Confused deputy attacks allow a compromised MCP server to acquire and replay OAuth tokens that OpenClaw has legitimately obtained for enterprise services [26].

Critical MCP vulnerabilities included CVE-2025-49596 (CVSS 9.4) in the Anthropic MCP Inspector, CVE-2025-66416 and CVE-2025-66414 in official MCP SDKs, CVE-2025-59159 (CVSS 9.7) in SillyTavern, CVE-2025-6514 (CVSS 9.6) in the mcp-remote package with 437,000 downloads, three chained RCE vulnerabilities in Anthropic's own mcp-server-git (CVE-2025-68143, CVE-2025-68144, CVE-2025-68145), and CVE-2026-25536 in the MCP TypeScript SDK enabling cross-client data leakage [24][27][28][29]. The vulnerability class that ClawJacked represented — local AI services trusting connections based on source address — proved endemic across the broader ecosystem, simultaneously affecting multiple MCP implementations and AI agent platforms. For enterprise OpenClaw deployments, MCP is both essential and risk-laden: the enterprise best practice is not to avoid MCP, but to treat every MCP server connection as an integration requiring the same security review, authentication enforcement, and network boundary controls applied to any third-party API integration.

Part II: Enterprise Adoption Framework

6. The Industrialization of Agentic Systems

NVIDIA's announcement of NemoClaw at GTC on March 16, 2026 marks a structural shift in which the dominant provider of AI compute infrastructure has begun formalizing agent orchestration patterns. When the world's most valuable technology company moves into the orchestration layer, it confirms that agentic systems have crossed the threshold from experimental to operational. This simultaneously validates enterprise investment in agentic AI and accelerates the adoption timeline faster than most governance programs can adapt. The NemoClaw stack provides three core components that directly target the OpenClaw security crisis: the OpenShell Runtime for kernel-level sandboxing using Landlock, seccomp, and network namespaces in a deny-by-default posture; an out-of-process policy engine expressed in declarative YAML that the agent cannot override even if fully compromised; and a privacy router that intercepts every inference request, classifies its data sensitivity, and routes queries containing personally identifiable information or proprietary data to locally-hosted Nemotron models rather than cloud endpoints [30][31].

The architectural significance of OpenShell lies in its out-of-process enforcement model. In OpenClaw's native permission model, permission checks occur within the agent's own process space, where a sufficiently sophisticated attack can modify, disable, or route around them. By enforcing policy in a separate process that the agent cannot access, terminate, or signal, OpenShell eliminates this attack surface. Filesystem and process constraints are locked at session creation time and cannot be relaxed mid-session, while network policies support hot-reload for operational flexibility [32][33]. NVIDIA's launch partner roster — including Box, Cisco, Atlassian, Salesforce, SAP, Adobe, CrowdStrike, Cohesity, and ServiceNow — reflects an intentional strategy to establish NemoClaw as enterprise integration infrastructure rather than a standalone security tool [30].

Regardless of its technical specifics, NVIDIA's entry creates three immediate pressures on enterprises. The first is executive-level validation: boardrooms will interpret NVIDIA's investment as confirmation that agentic systems are strategically important and that delayed adoption carries opportunity cost. The second is acceleration of shadow adoption: announcements of this magnitude increase developer experimentation and expand the attack surface before controls are in place. The third is vendor-led architecture bias: enterprises will be presented with integrated, opinionated stacks that offer operational simplicity in exchange for reduced flexibility and implicit trust assumptions. What NemoClaw does not solve remains critically important: it does not inherently address agent identity and authentication

standards, supply chain trust for agent skills and tools, prompt injection and behavioral manipulation risks, cross-agent communication governance, or regulatory accountability and auditability. Malicious ClawHub skills will execute within a NemoClaw sandbox just as they would without one, albeit with reduced blast radius. NemoClaw reduces blast radius; it does not provide governance.

7. Decision Framework: OpenClaw vs. Alternatives

Before proceeding to the adoption framework, organizations should evaluate whether OpenClaw is the appropriate platform for their use case. The decision about platform selection is logically prior to the question of how to secure it, and for some use cases managed alternatives may provide stronger native security properties at lower total cost of ownership.

Dimension	OpenClaw + NemoClaw	AWS Bedrock AgentCore	Azure AI Agent Service	Google ADK	Claude Cowork	OpenAI Codex
Execution isolation	NemoClaw OpenShell (kernel sandbox)	Cloud container, managed	Cloud container, managed	Cloud container, managed	VM via Apple Virtualization / hypervisor	Secure cloud container, internet disabled
Model flexibility	Any model (local + cloud)	Bedrock models + custom	Azure OpenAI + custom	Gemini + custom	Claude only	OpenAI models only
Identity integration	Manual (OAuth, SPIFFE)	Native IAM integration	Entra ID native	Google IAM native	Enterprise admin controls	Workspace controls
Compliance certifications	None (open source)	SOC 2, HIPAA, FedRAMP	SOC 2, HIPAA, ISO 27001	SOC 2, HIPAA	SOC 2 (Team/Enterprise)	SOC 2 (Enterprise)
Supply chain governance	Manual skill vetting	Managed tool registry	Managed tool registry	Managed tool registry	Curated integrations	Curated tools
Cost model	Infrastructure + security ops	Per-invocation	Per-invocation	Per-invocation	Per-seat subscription	Per-seat subscription

Dimension	OpenClaw + NemoClaw	AWS Bedrock AgentCore	Azure AI Agent Service	Google ADK	Claude Cowork	OpenAI Codex
Extensibility	Highest (open skill ecosystem)	Moderate (managed tools)	Moderate (managed tools)	Moderate (managed tools)	Limited (curated)	Limited (curated)

For desktop agent use cases where execution isolation is the primary concern, Claude Cowork provides a stronger isolation boundary through hypervisor-enforced virtual machines. For coding agent use cases, OpenAI Codex operates in secure cloud containers with internet access disabled by default during task execution, eliminating the entire class of exfiltration attacks relying on outbound connections [34][35].

Dimension	OpenClaw	LangChain / LangGraph	CrewAI	AutoGen (Microsoft)
Primary use case	Personal/desktop AI agent	Application agent pipelines	Multi-agent collaboration	Multi-agent conversation
Extensibility model	Skill registry (ClawHub)	Python packages + tools	Agent role definitions	Agent chat framework
Security track record	Multiple critical CVEs (2026)	Prompt injection concerns; fewer CVEs	Newer; limited security research	Microsoft-backed; moderate scrutiny
Enterprise adoption	High (shadow IT driven)	High (developer-driven)	Growing	Growing
Community size	250K GitHub stars	105K+ GitHub stars	25K+ GitHub stars	35K+ GitHub stars
NHI / IAM support	None native	None native	None native	Azure AD integration

OpenClaw's unique value proposition is its accessibility, community ecosystem, and model-agnostic desktop agent capabilities. Its unique liability is the combination of that accessibility with architectural security deficits that have been more thoroughly exploited than any competing framework. Some organizations will correctly conclude that paying for a managed platform is less expensive than building the security infrastructure described in subsequent sections — particularly organizations without existing security automation maturity.

8. The Agentic Standards Landscape

The simultaneous maturation of agentic communication standards is reshaping the interoperability surface that any OpenClaw deployment must integrate with. Anthropic's Model Context Protocol has become the de facto standard for connecting AI agents to external tools and data sources, but its security posture remains a significant enterprise concern. The current specification (2025-11-25) introduced Streamable HTTP transport and a draft specification (2025-06-18) added an OAuth 2.1 security overhaul that classifies MCP servers as OAuth Resource Servers with mandatory RFC 8707 Resource Indicators [36]. Despite these improvements, the concentrated wave of 30+ CVEs in 60 days demonstrates that the ecosystem has outgrown its security design.

Google's Agent-to-Agent (A2A) protocol, now at version 0.3 and donated to the Linux Foundation, provides the emerging standard for inter-agent discovery, authorization, and communication. Built on HTTP, SSE, and JSON-RPC, A2A supports dynamic negotiation with propose/accept/counter-offer patterns, standardized cryptographic identity via security card signing, and gRPC support for high-throughput deployments [37][38]. The protocol has attracted over 150 supporting organizations and absorbed IBM's Agent Communication Protocol under the Linux Foundation umbrella. CopilotKit's Agent-User Interaction Protocol (AG-UI) addresses the gap that neither MCP nor A2A covers: how agents communicate with human-facing interfaces. AG-UI defines approximately 16 standard event types streamed as JSON over HTTP or WebSocket, covering messages, tool calls, state patches, and lifecycle signals [39]. Amazon Bedrock AgentCore Runtime added AG-UI support on March 13, 2026 [40].

The emerging consensus organizes agentic protocols into functional layers: A2A and ANP at the networking layer for agent-to-agent communication and internet-scale discovery; MCP at the context layer for agent-to-tool integration; and AG-UI at the presentation layer for human interaction [41]. Enterprise OpenClaw deployments should be designed with awareness of this layering model because it creates natural enforcement points for security controls at each layer. However, organizations should also recognize that this landscape is fragmented, immature, and changing rapidly, and should implement protocol-layer abstractions that allow switching between protocol versions as the landscape consolidates rather than hard-coding against specific implementations.

9. Enterprise Adoption Maturity Framework

The emergence of platforms such as NemoClaw and the broader industrialization of agentic systems reinforce a central premise: what enterprises face is not a tooling problem but a control plane problem. Execution layers will evolve rapidly as infrastructure providers compete and open-source innovation continues. The durable requirement — the one that persists regardless of which agent framework prevails — is a governance architecture that provides identity governance, authorization enforcement, data control, and behavioral observability across all agent deployments.

The following framework defines four phases of OpenClaw adoption, each with specific security prerequisites that must be satisfied before advancing. Two capabilities are cross-cutting requirements at every phase: incident response (organizations need agent compromise procedures from Phase 0, because shadow IT instances discovered today can be compromised today) and monitoring (agent activity must be observable at every phase, with sophistication increasing as deployment scope expands).

Phase 0: Discovery and Containment. Most enterprises reading this paper already have OpenClaw running on developer workstations without IT knowledge. Endpoint detection tools should be configured to identify running OpenClaw processes, listening WebSocket ports (default 18789), and the presence of the `~/.clawdbot` or

`~/.openclaw` directory structures. Any instance running a version prior to v2026.2.25 should be treated as actively vulnerable and updated or quarantined immediately. Rather than issuing a blanket prohibition, organizations should communicate a clear timeline for managed adoption.

Phase 0 Control	Implementation	Validation
Endpoint discovery	EDR query for OpenClaw processes and directories	Inventory covers all developer endpoints
Version verification	Automated check against minimum v2026.2.25	No instances below minimum version
Skill audit	Compare installed skills against ClawHavoc IOC list	All flagged skills removed or quarantined
Network visibility	Monitor for Moltbook heartbeat and update traffic	Outbound connections documented
Policy communication	Written timeline for managed adoption path	Acknowledged by development leadership
Baseline IR checklist	One-page SOC response procedure for agent compromise	SOC team briefed and checklist accessible

Phase 1: Sandbox Experimentation. Phase 1 enables controlled experimentation in architecturally isolated environments. Organizations with NVIDIA hardware should deploy NemoClaw's OpenShell Runtime; others should provision dedicated virtual machines or container environments with no network connectivity to production VPCs. The sandbox should contain only synthetic data, development-grade API keys, and dummy credentials. Skill installation should be limited to a curated allowlist, and MCP server connections should be restricted to locally hosted servers running within the same isolation boundary.

Phase 1 Control	Implementation	Validation
Execution isolation	NemoClaw OpenShell or dedicated VM/container	No host filesystem access from agent
Network segmentation	Firewall rules blocking production VPC access	Penetration test confirms no production reachability
Data isolation	Synthetic/public data only in sandbox	No production data accessible to agent
Skill allowlist	Curated, security-reviewed skill set	Only allowlisted skills installable
MCP server pinning	Locally hosted, version-pinned MCP servers	No connections to production MCP endpoints

Phase 1 Control	Implementation	Validation
Sandbox identity	Dedicated service accounts, no production IdP	Credentials verified as non-production
Monitoring	Agent activity logs forwarded to SIEM	Log ingestion verified

Phase 2: Controlled Pilot Deployment. Phase 2 introduces OpenClaw to real workflows with real data under controlled conditions. The critical new requirement is identity architecture: agents must be integrated with the organization's IAM infrastructure using least-privilege scoping with ephemeral credentials (5-15 minute TTL), automatically revoked upon task completion. Human-in-the-loop controls must be implemented for irreversible actions. Insider threat controls become relevant because agents now access real data, requiring skill installation logging, outbound data volume monitoring, and policy-based restrictions on agent-accessible data classifications.

Phase 2 Control	Implementation	Validation
IAM integration	Enterprise IdP, ephemeral credentials (5-15 min TTL)	No standing access tokens in agent configuration
Least-privilege scoping	Per-task credential issuance with minimum permissions	Permission audit confirms no excess grants
Human-in-the-loop	AG-UI approval workflows for irreversible actions	Destructive actions blocked without human approval
Comprehensive audit logging	Immutable, centralized logs for all agent actions	Log coverage verified via agent activity replay
Behavioral monitoring	Anomaly detection on tool invocation patterns	Alert generation confirmed for test anomalies
Insider threat controls	Skill installation logging, outbound data monitoring	Exfiltration-pattern alerts configured
Skill security review	Static analysis, permission audit, network audit	Review gate blocks unapproved skill installation
MCP server governance	Internal registry, versioned deployment, change management	No direct public pulls in production

Phase 3: Production Hardening. Phase 3 represents full production deployment requiring demonstrated stability over a sustained period — typically 30+ days of monitored operation without security incidents. Zero Trust architecture must be fully applied: every agent action authorized against current policy at execution time, not at token issuance time.

Cross-boundary delegation must be governed through explicit delegation contracts for every integration point. NemoClaw's privacy router should be configured with explicit data classification policies enforced at the infrastructure layer.

Phase 3 Control	Implementation	Validation
Zero Trust agent authorization	Per-action policy evaluation at execution time	Policy bypass attempts detected and blocked
Cross-boundary delegation contracts	Documented identity, permissions, revocation for each integration	All A2A and external API integrations covered
Privacy router configuration	Data classification policies enforced at infrastructure layer	Sensitive data confirmed as local-only processing
Supply chain signing	Internal registry, code-signed skills, vulnerability scanning	No unsigned or unreviewed skills in production
Agent incident response	Documented playbook with sub-5-minute containment target	Tabletop exercise completed successfully
Continuous compliance	Automated control validation against CCM and AICM	Compliance dashboard operational

Part III: Zero Trust Architecture for Agentic AI

10. Why Zero Trust Is Necessary for Agentic AI

The perimeter security model that governed enterprise computing for three decades rested on five foundational premises, each of which agentic AI fundamentally invalidates. The first premise held that entities inside the network boundary could be granted elevated trust; agentic AI systems operate inside the network but process instructions and content from sources entirely outside the organization's control. The second premise assumed that software behavior was deterministic and auditable; agentic AI systems are fundamentally non-deterministic, with the same prompt potentially producing different tool invocations depending on the language model's interpretation, conversation history, and stochastic sampling parameters. The third premise expected that privileged operations would be mediated by human operators; agentic AI systems are designed specifically to operate without continuous human oversight. The fourth premise assumed that network communication patterns were knowable and constrainable; agents communicate across multiple protocols and endpoints in patterns that shift based on their tasks. The fifth premise held that access rights remained appropriate for the duration of a session; agents persist across sessions, accumulate context, and expand their effective capabilities through delegation chains.

Analysis of 10 major agentic AI incidents documented in CSA's research reveals a consistent failure pattern: implicit trust granted without continuous verification. Every significant compromise traces to a point where an agent, tool, skill, or communication channel was trusted by default rather than verified at each interaction.

Incident	Trust Assumption Violated	Impact	Zero Trust Control
ClawHavoc (800+ malicious skills)	Skills from public registry are safe	Credential theft across thousands of deployments	Supply chain verification, code signing, internal registries
CVE-2026-25253 (RCE)	Gateway WebSocket connections are trusted	Remote code execution via auth token theft	Origin validation, per-action authorization, token scoping
ClawJacked (WebSocket hijack)	Local WebSocket connections are trusted	Agent session hijack via DNS rebinding	Origin validation, SDP, network isolation
Agent Commander (C2)	Agent instructions come from authorized sources	Full command-and-control via prompt injection	Input verification, behavioral monitoring, autonomy limits
Clinejection (npm poisoning)	Package metadata is benign	Development agent hijack through dependency descriptions	Content sanitization, tool output validation
Alibaba ROME (crypto mining)	Autonomous agents stay on-task	Unauthorized GPU diversion and cloud resource consumption	Autonomy governance, continuous behavioral verification
Salesloft Drift (700+ orgs)	Standing OAuth tokens remain appropriate	Cross-tenant data exposure via bulk SOQL exports	JIT credentials, session-scoped access, continuous verification
Islands of Agents (delegation chains)	Delegated agents inherit appropriate trust	Privilege escalation through multi-hop delegation	Delegation scope narrowing, composite autonomy assessment
PerplexedBrowser (browser agent hijack)	Web content rendered by agents is safe	Data exfiltration through invisible web elements	Content isolation, output filtering, behavioral bounds
CyberStrikeAI (offensive AI)	AI agents are used defensively	600+ devices compromised in autonomous pen test	Autonomy caps, human-in-the-loop gates, scope constraints

11. CSA Zero Trust Principles Applied to Agents

CSA's 11 Zero Trust Guiding Principles, published in version 1.1, provide the foundational philosophy that underlies all Zero Trust implementations [42]. These principles were developed in the context of traditional enterprise IT, but their formulation is sufficiently abstract that they apply directly to agentic AI governance without fundamental modification. "Begin with the End in Mind" instructs organizations to identify the protect surfaces — the critical data, applications, assets, and services — that agents interact with. "Access Is a Deliberate Act" demands that every tool invocation, data access, inter-agent delegation, and autonomy level transition is individually authorized based on current context, not inherited from past authorizations. "Inside Out, not Outside In" redirects security from protecting the network perimeter to protecting the resources themselves — an agent running inside a corporate network is no more trustworthy than one running in the cloud.

"Breaches Happen" accepts that compromise is inevitable and designs for containment, demanding that every agent deployment assume at least one agent in the ecosystem will be compromised and bound the blast radius through architectural controls. "Never Trust, Always Verify" demands that agent identity is verified at every interaction. "Least Privilege" requires that agents receive only the permissions necessary for their current task. "Microsegmentation" demands that MCP, A2A, and AG-UI channels are isolated from each other and from general network traffic. "Continuous Verification" requires ongoing behavioral monitoring that detects when an agent's actions diverge from expected patterns, even if identity credentials remain valid. Zero Trust is not merely a technical architecture for agent governance; it is a security philosophy that informs organizational decisions about autonomy, automation, error handling, and oversight — protecting organizations not only from adversarial attacks but from emergent behavior failures that arise from excessive trust in agent capabilities.

12. The Five-Step Zero Trust Methodology for Agents

CSA's established five-step Zero Trust methodology provides a systematic implementation framework when adapted to the specific requirements of agentic AI security.

The first step defines the protect surfaces: agent configurations (system prompts, MCP server connections, credential bindings, autonomy settings), credentials managed by or accessible to agents (API keys, database connection strings, OAuth tokens, SPIFFE identities), data accessed during task execution (which shifts dynamically based on the current task), communication channels (MCP WebSocket connections, A2A sessions, AG-UI rendering pipelines), and tool connections that define the agent's operational capabilities. The second step maps all transaction flows crossing these protect surfaces across four primary types: agent-to-tool flows traversing MCP, agent-to-agent flows traversing A2A with delegation scope tracking, agent-to-user flows traversing AG-UI including human-in-the-loop approval points, and agent-to-data flows representing direct database access, file system operations, and API calls.

The third step builds the Zero Trust architecture through three control layers: per-action verification ensuring every operation is individually authorized, boundary enforcement establishing isolation between agents, resources, and communication channels, and delegation controls governing authority transfer between agents. The fourth step creates policies across three dimensions: autonomy-level policies defining permitted operations at each level, AICM policies mapping technical controls to protect surfaces, and boundary specification policies defining the precise scope of each agent's operational authority. The fifth step establishes continuous monitoring and maintenance: behavioral monitoring

tracks each agent against established baselines, dynamic autonomy adjustment modifies autonomy levels in real time based on verification results, and agents exhibiting anomalous behavior are automatically reduced to lower autonomy levels requiring more frequent human oversight.

13. Zero Trust Identity for AI Agents

The identity management challenge for OpenClaw deployments reflects a broader enterprise crisis: non-human identities outnumber human identities at a ratio of 45 to 1, 78 percent of organizations lack documented policies for AI identity creation and removal, and 44 percent rely on static API keys as their primary agent authentication method [43] [44]. OpenClaw compounds this challenge because each installation requires credentials for at least one model provider API and potentially multiple MCP servers, enterprise services, and code repositories — credentials stored locally and accessible to any attacker who achieves influence over the agent.

CSA's Islands of Agents framework identifies four agent categories — personal desktop, coding, SaaS, and enterprise — each requiring different identity treatment [45]. The fundamental architecture decision for each is whether the agent operates as the user, on behalf of the user (receiving a delegated, scoped subset of permissions), or as its own entity (holding an independent machine identity). The first option produces maximum blast radius upon compromise; the second is appropriate for Phase 2 pilots; the third is appropriate for Phase 3 production.

The Agentic Trust Framework establishes a tiered trust model mapped to enterprise roles: Intern (new, unverified — read-only access to non-sensitive resources with frequent human oversight), Associate (moderate verification — broader access with periodic checkpoints), Manager (extensively verified — delegation-capable with behavioral monitoring), and Principal (certified — high autonomy within defined scope, redundant monitoring). Trust tier transitions are governed by explicit policy and verified by behavioral monitoring, ensuring no agent receives elevated trust without demonstrated trustworthiness.

For workload identity, SPIFFE/SPIRE provides the most mature approach in cloud-native environments, assigning each agent a cryptographic identity (SVID) attested by the deployment platform rather than configured by an administrator. Cloud IAM services — Azure Entra workload identities, AWS IAM roles for service accounts, Google Cloud Workload Identity Federation, and Okta's Agent Identity Framework — provide alternatives for cloud-native deployments [46][47]. JIT credential provisioning replaces standing credentials: when an agent needs to access a resource, it presents its identity to a secrets manager (HashiCorp Vault, AWS Secrets Manager, Azure Key Vault) which issues a scoped credential with a short TTL, typically 5-15 minutes. A2A Security Cards provide standardized formats for cryptographically signed agent identity in cross-platform communication [37].

14. Zero Trust Network and MCP Security

CSA's Software-Defined Perimeter specification provides the network architecture foundation for Zero Trust agent communication. SDP implements a "dark network" approach using Single Packet Authorization (SPA), where MCP servers and backend resources are invisible to unauthorized entities and connections are established only after identity verification. This directly addresses the ClawJacked vulnerability class: DNS rebinding attacks succeed because the target service is network-addressable and accepts connections based on network-level access rather than cryptographic identity; in an SDP architecture, the endpoint is invisible until authenticated through SPA [42].

Microsegmentation isolates MCP traffic (agent-to-tool), A2A traffic (agent-to-agent), and AG-UI traffic (agent-to-user) into separate network segments with independent access controls. All channels employ mutual TLS authentication using SPIFFE SVIDs, eliminating attacks that exploit unauthenticated or one-way authenticated channels.

Zero Trust for MCP tool invocations is enforced through a policy-as-code interceptor that sits between the agent and every MCP server, evaluating each invocation against identity, scope, parameters, rate, and behavioral context before permitting execution. Policies are expressed in OPA Rego, Cedar, or equivalent declarative languages and version-controlled alongside agent configurations [48]. The interceptor performs deep parameter inspection for data loss prevention and injection detection, scans for sensitive data patterns, and blocks invocations that would send classified data to unauthorized tools. Tool description integrity is enforced through cryptographic signing: each description is signed by the publisher, compared against a known-good baseline, and any modification triggers a security alert — the signature of a rug pull attack. MCP tools executing code run in isolated environments with container sandboxing, filesystem isolation, network isolation, and resource limits preventing a compromised tool from affecting the MCP server, other tools, or the agent's host system.

15. Agent-to-Agent Delegation and Autonomy Governance

When Agent A delegates a task to Agent B, Agent A transfers a portion of its authority and trusts Agent B to exercise it appropriately. Zero Trust makes this transfer explicit, scoped, verified, and monitored. A fundamental principle is that each delegation hop reduces available authority, never expands it: if Agent A has read-write database access and delegates a query task to Agent B, Agent B receives read-only access to specific tables; if Agent B further delegates to Agent C, Agent C's access is narrower still. The delegation framework generates scoped credentials for each hop, maintains chain records for audit, and calculates composite autonomy levels to prevent implicit escalation through multi-agent chains [45][49].

CSA's Autonomy Levels framework defines five levels, mapped to Zero Trust controls as follows:

Autonomy Level	Human Oversight Model	ZT Credential Model	Behavioral Monitoring	Appropriate Use Cases
Level 0 (Fully Supervised)	Every action requires approval	Session-scoped, provisioned per approval	Complete logging of all actions	Unfamiliar environments, high-sensitivity data
Level 1 (Human-Guided)	Approved categories autonomous; exceptions escalate	Category-based tokens with anomaly triggers	Out-of-category anomaly detection	Routine, well-understood tasks
Level 2 (Semi-Autonomous)	Periodic checkpoints (e.g., every 15 min / 50 ops)	Scope-constrained with auto-narrowing on anomaly	Behavioral baseline validation	Mature deployments with known patterns

Autonomy Level	Human Oversight Model	ZT Credential Model	Behavioral Monitoring	Appropriate Use Cases
Level 3 (Highly Autonomous)	Minimal oversight; gates at organizational boundaries	Delegation-capable with mandatory scope narrowing	Continuous verification, tight anomaly thresholds	Extensive operational history, explicit certification
Level 4 (Fully Autonomous)	No routine oversight within defined scope	Architecturally bounded (unauthorized actions physically impossible)	Zero-tolerance thresholds, redundant systems	Exceptional circumstances only; robust containment required

Zero Trust reframes prompt injection from a technical vulnerability (a failure to sanitize input) to an autonomy governance violation: an unauthorized attempt to modify the agent's operational objectives. Rather than relying solely on input sanitization, which is extremely difficult to make comprehensive against adversarial natural language, Zero Trust defends through defense-in-depth across all control layers. Input validation catches known patterns; behavioral monitoring detects divergence from expected actions; scope constraints prevent operations outside authorized bounds; delegation controls prevent a compromised agent from passing compromise to others; and output filtering prevents exfiltration even under adversarial instruction.

Part IV: Operational Hardening

16. Installation and Configuration Security

The first hardening decision occurs before OpenClaw executes: where the installation binary originates and how its integrity is verified. Organizations should download exclusively from the official GitHub release page or a vetted package manager, verify the SHA-256 cryptographic signature of every artifact, maintain an internal mirror of approved versions, and block installation from unauthorized sources through endpoint management policies. Automatic updates should be disabled in enterprise deployments; instead, updates should flow through a controlled pipeline where the security team evaluates each release, tests it in staging, and deploys approved versions through the organization's software distribution infrastructure [5].

OpenClaw's default configuration prioritizes usability over security. The following baseline hardening measures represent the minimum viable posture: disable auto-approval of tool invocations requiring explicit user confirmation for any action modifying the file system, executing shell commands, or accessing sensitive data; restrict the default working directory to a designated workspace rather than the entire home directory; disable or restrict browser automation to a curated domain allowlist unless specifically required; enforce Gateway authentication using token mode with a minimum 32-character random token, disabling the localhost trust exemption that enabled ClawJacked; disable mDNS broadcasting with `discovery.mdns.mode: "off"` or `OPENCLAW_DISABLE_BONJOUR=1`; restrict shell execution by setting `tools.exec.security` to `"deny"` and `tools.exec.ask` to `"always"`; deny

dangerous tool groups including `group:automation`, `group:runtime`, `group:fs`, and individual control-plane tools (`gateway`, `cron`, `sessions_spawn`, `sessions_send`); configure session timeouts with `session.dmScope` set to `"per-channel-peer"` to prevent context leakage; and lock down SOUL.md and HEARTBEAT.md files to read-only with administrator ownership.

17. Skill Vetting and Supply Chain Integrity

Organizations should implement a formal four-stage skill vetting pipeline that treats every ClawHub skill as untrusted code until evaluated and approved. Stage 1 performs source code review including static analysis for embedded prompt injection payloads, network communication patterns, data exfiltration indicators, and credential access attempts; automated scanning for known malware signatures, obfuscated code, encoded payloads, and attacker infrastructure references. Stage 2 conducts behavioral sandbox testing in an isolated environment that monitors all file system access, network connections, process creation, and inter-process communication, flagging any access to credential stores, SSH keys, browser data, or cryptocurrency wallets. Stage 3 assigns minimum permissions required for each skill's intended function — a code formatting skill should not have network access; a documentation search skill should not have write access. Stage 4 provides ongoing monitoring because rug pull attacks can introduce malicious behavior in subsequent updates, requiring every skill update to re-enter the pipeline at Stage 1 [12][26].

Organizations should maintain an internal registry of approved skills, versioned and cryptographically hashed, that serves as the authoritative source for skill installation. OpenClaw should be configured to install skills only from this registry, not directly from ClawHub. Several open-source vetting tools should be integrated into the pipeline: Clawdex (Koi Security) scans against a continuously updated malicious entry database; Cisco DefenseClaw and Skill Scanner combines static analysis, behavioral analysis, LLM-assisted semantic inspection, and VirusTotal scanning; Snyk mcp-scan scans MCP server configurations for known vulnerabilities; Prompt Security ClawSec provides automated deployment scanning; and Knostic openclaw-telemetry captures tool calls with JSONL output and SIEM forwarding [11][13][50].

For MCP server governance, organizations should implement tool description integrity verification through cryptographic hashing of approved descriptions, blocking any tool invocation when descriptions do not match approved baselines. Model supply chain controls should include version pinning to prevent unexpected behavioral changes, integrity verification of model artifacts against known-good hashes, and privacy routing through NemoClaw's classification policies for cloud-hosted model connections.

18. Runtime Sandboxing and Isolation

Runtime sandboxing limits blast radius when other controls fail. No amount of skill vetting or configuration hardening can eliminate the risk of prompt injection — a structural property of instruction-following architectures that mitigations can raise the cost of exploiting but cannot prevent entirely. Organizations deploying Docker-based isolation should apply rigorous hardening: use a minimal base image with only required binaries; drop all Linux capabilities with `--cap-drop=ALL`; enable read-only root filesystem with `--read-only` and explicit writable mount points only for the working directory and temporary files; apply a custom seccomp profile restricting available system calls; isolate the

container network with explicit egress rules allowing only approved endpoints; bind-mount only required directories using read-only mounts wherever possible, never mounting the entire home directory, SSH directory, browser profile, or cloud configuration; and run as a non-root user inside the container [10].

NemoClaw's OpenShell Runtime provides purpose-built security addressing the limitations of manual Docker hardening. Its kernel-level sandboxing combines Landlock for file system access control, seccomp for system call filtering, and network namespaces for network isolation in a deny-by-default posture fundamentally different from Docker's default-allow model. The out-of-process policy engine ensures that a prompt injection attack gaining control of the agent's reasoning cannot disable or circumvent policy enforcement. The privacy router mediates all outbound inference requests, stripping or redacting sensitive content before routing and logging every decision for compliance auditing [30][32]. For advanced deployments, NemoClaw supports a reader-actor separation pattern: one sandboxed runtime with low authority processes external content and extracts structured data, while a separate runtime with restricted permissions executes approved actions based on sanitized output, containing indirect injection damage to the reader instance.

Operating system-level controls provide defense-in-depth regardless of sandbox deployment. On macOS, enable the application sandbox through MDM configuration profiles, restrict accessibility features, and use TCC policies for explicit approval of camera, microphone, file system, and network access beyond designated working directories. On Linux, deploy AppArmor or SELinux mandatory access control profiles denying access to sensitive directories including ~/.ssh, ~/.gnupg, ~/.aws, ~/.config/gcloud, browser credential stores, and cryptocurrency wallet directories. On Windows, use AppLocker or Windows Defender Application Control policies to restrict launchable executables, and use Windows Sandbox or Hyper-V isolation for maximum containment.

19. File System, Network, and Credential Controls

OpenClaw's file system access is both its primary value mechanism and its primary damage mechanism upon compromise. The principle is straightforward: access to the minimum set of files and directories required for the current task, and no more. A dedicated workspace directory should be defined for each deployment, with the agent configured to treat it as root. The following directories should be excluded from OpenClaw's access through both application configuration and operating system controls:

Directory	Content	Risk
~/.ssh/	SSH private keys	Remote system access
~/.gnupg/	GPG keys, trust databases	Cryptographic identity theft
~/.aws/, ~/.config/gcloud/	Cloud provider credentials	Cloud infrastructure compromise
~/.kube/	Kubernetes configurations and tokens	Container orchestration access
Browser profile directories	Saved passwords, cookies, sessions	Account takeover
~/Library/Keychains/ (macOS)	System keychain	Credential store access

Directory	Content	Risk
~/.password-store/	Pass password manager	Password vault exfiltration
Cryptocurrency wallet directories	Private keys	Financial theft
~/.docker/config.json	Docker registry credentials	Container registry access
~/.npmrc, ~/.pypirc	Package registry tokens	Supply chain compromise

Network egress controls limit where OpenClaw can send data, reducing the value of any data an attacker causes the agent to collect. Egress allowlisting should restrict access to model provider API endpoints, approved MCP servers, and legitimate additional services, blocking all other outbound connections. DNS filtering should block queries for known malicious domains, newly registered domains, and dynamic DNS services, because prompt injection attacks frequently use URL-based exfiltration encoding data in DNS queries or URL parameters. All traffic should route through an inspecting proxy applying DLP policies and logging requests. In NemoClaw deployments, the OpenShell network namespace enforcement provides kernel-level egress control that is more robust than application-level alternatives because it operates below the level at which the agent can influence network behavior [32].

For credential management, organizations should eliminate static API keys in favor of short-lived, automatically rotating credentials using OAuth-based authentication or workload identity federation. Credentials requiring local storage should be placed in the operating system's credential manager rather than environment variables or plaintext files. API keys should be scoped to minimum capabilities with per-key rate limits reflecting expected usage patterns. Every OpenClaw installation should be traceable to a specific human sponsor through a registry recording the sponsoring employee, approved use case, assigned credentials, granted permissions, and monitoring configuration [43][44].

SOUL.md protection requires filesystem ACLs restricting write access to administrative accounts, file integrity monitoring (FIM) configured to watch SOUL.md, HEARTBEAT.md, and the entire `~/.openclaw/` configuration directory with immediate alerts on any modification, and git-versioned backups enabling rapid comparison against known-good baselines.

20. Behavioral Monitoring, Detection, and Incident Response

Traditional user behavior analytics identifies anomalies by comparing current activity against a baseline of normal behavior. Agent monitoring faces a structural challenge that UBA was not designed for: a compromised agent exercises the same permissions, calls the same APIs, and generates the same log entries as a legitimately operating agent. The attack looks like work. Effective agent monitoring must therefore look beyond access patterns to behavioral indicators distinguishing legitimate task execution from adversarial manipulation: instruction-data boundary violations where the agent executes actions appearing to originate from processed content rather than user instructions; credential access patterns not required for the stated task; unexpected network destinations not on approved allowlists; SOUL.md modifications outside administrative change management; unexpected skill or MCP configuration changes; data staging behavior inconsistent with the stated task (copying, compressing, encoding data as pre-exfiltration activity); and temporal anomalies where the agent executes actions during periods when the sponsoring user is not actively interacting with it [21].

OpenClaw deployments should emit structured telemetry capturing every tool invocation and its parameters, every file read and write operation, every network connection and destination, every shell command executed, every MCP server interaction and tool call, every skill invocation, every configuration modification, the user prompt initiating each action chain, and the model's reasoning trace for each decision.

Detection Rule	Description	Severity	ATLAS Technique
SOUL.md Modified	File hash changed outside change management	Critical	AML.T0081
Credential Store Access	Agent accessed ~/.ssh, Keychain, or browser credentials	High	AML.T0082, AML.T0083
Unexpected Egress	Outbound connection to non-allowlisted destination	High	AML.T0086
Skill Config Changed	Skill manifest or MCP config modified at runtime	High	AML.T0081
Bulk File Read	Agent read >50 files in 60 seconds outside working directory	Medium	AML.T0085
Off-Hours Activity	Agent active during non-business hours without user session	Medium	—
Encoded Data Output	Agent produced base64/hex-encoded content in network request	Medium	AML.T0086
New MCP Server	Agent connected to previously unseen MCP server	Medium	AML.T0099
Shell Escalation	Agent executed sudo, su, or privilege escalation command	Critical	AML.T0105
Package Installation	Agent installed system packages or pip/npm packages	Medium	AML.T0104

Incident response for a compromised OpenClaw installation requires three specific playbooks. For malicious skill detection, the response includes immediate removal from all installations, identification of all systems where the skill was active, review of activity logs, rotation of any accessible credentials, endpoint scanning for dropped malware, and registry update to block future installation. For SOUL.md persistence attacks, the response includes immediate quarantine, comparison against baseline to identify injected instructions, review of post-modification agent activity, identification of the modification vector, restoration from trusted template, and investigation of secondary persistence mechanisms. For data exfiltration via prompt injection, the response includes immediate network isolation, identification of accessed and transmitted data, determination of the injection source, notification of affected data owners with breach notification procedures, and enhancement of content filtering rules.

Organizations with OpenClaw already in production should follow a time-boxed emergency action plan. Day 0 actions (hours): rotate all standing credentials, restrict MCP connections to a strict allowlist, disable public skill installation from ClawHub, enable full tool invocation logging, and block unapproved outbound traffic. Day 30 actions (weeks): deploy JIT credential provisioning through a secrets management platform, introduce an MCP proxy enforcement layer, establish SPIFFE or cloud IAM workload identity for all agents, and segment network traffic by protocol type. Day 90 actions (months): enforce per-action authorization for all tool invocations, implement delegation scope controls across agent chains, deploy behavioral monitoring with dynamic autonomy adjustment, operationalize internal skill registry with code signing, and formalize autonomy governance with graduated response capabilities.

Part V: Framework Alignment and Governance

21. Control Framework Mapping

Each hardening domain addresses one or more OWASP Top 10 for Agentic Applications risk categories, MITRE ATLAS agent techniques, MAESTRO threat modeling layers, and AICM control domains.

OWASP Risk	Hardening Domain	Key Controls
ASI01: Agent Goal Hijack	Sections 16, 18, 20	SOUL.md integrity, sandboxing, behavioral monitoring
ASI02: Tool Misuse and Exploitation	Sections 17, 14	Skill vetting, MCP server governance, permission scoping
ASI03: Identity and Privilege Abuse	Sections 13, 19	Credential hardening, least privilege, human sponsor tracing
ASI04: Agentic Supply Chain Vulnerabilities	Sections 16, 17, 14	Installation verification, skill registry, MCP allowlisting
ASI05: Unexpected Code Execution	Sections 18, 19	Docker hardening, NemoClaw sandboxing, OS-level controls
ASI06: Memory and Context Poisoning	Sections 16, 20, 4	SOUL.md lockdown, FIM, memory cleanup in IR
ASI07: Insecure Inter-Agent Communication	Section 14	MCP authentication, mutual TLS, tool description integrity
ASI08: Cascading Failures	Sections 18, 20	Sandbox containment, monitoring, circuit breaker patterns

OWASP Risk	Hardening Domain	Key Controls
ASI09: Human-Agent Trust Exploitation	Sections 20, 19	Behavioral monitoring, acceptable use policy, approval workflows
ASI10: Rogue Agents	Sections 18, 13, 20	Sandboxing, credential scoping, detection rules, IR playbooks

MITRE ATLAS conducted dedicated investigations of OpenClaw security incidents in February 2026, producing four case studies mapping documented attack chains to ATLAS techniques [51]. The hardening controls provide detection or mitigation for the following ATLAS techniques:

ATLAS Technique	Description	Hardening Control
AML.T0080	AI Agent Context Poisoning	SOUL.md integrity, FIM
AML.T0080.000	Memory Manipulation	Memory cleanup, session timeouts
AML.T0080.001	Thread Injection	Behavioral monitoring, sandboxing
AML.T0081	Modify AI Agent Configuration	SOUL.md lockdown, config FIM
AML.T0082	RAG Credential Harvesting	Credential isolation, file access controls
AML.T0083	Credentials from AI Agent Configuration	Credential store isolation, OS controls
AML.T0085.001	Data from AI Agent Tools	MCP governance, permission scoping
AML.T0086	Exfiltration via AI Agent Tool Invocation	Egress control, DLP, proxy enforcement
AML.T0099	AI Agent Tool Data Poisoning	Tool description integrity, MCP allowlisting
AML.T0100	AI Agent Clickbait	Browser restrictions, domain allowlisting
AML.T0101	Data Destruction via AI Agent Tool Invocation	Read-only mounts, backup verification
AML.T0104	Publish Poisoned AI Agent Tool	Skill vetting pipeline, internal registry
AML.T0105	Escape to Host	Docker hardening, NemoClaw sandbox, OS controls

OpenClaw's attack surface spans all seven MAESTRO layers:

MAESTRO Layer	OpenClaw Component	Hardening Domain
L1: Foundation Models	Model provider API connections	Credential management, egress control
L2: Data Operations	File access, RAG, persistent memory	File system controls, memory integrity
L3: Agent Frameworks	OpenClaw runtime, skill execution	Skill vetting, configuration hardening
L4: Deployment and Infrastructure	Docker/NemoClaw, OS, network	Sandboxing, OS controls, network controls
L5: Evaluation and Observability	Logging, monitoring, alerting	Behavioral monitoring, SIEM integration
L6: Security and Compliance	Governance, policy, compliance	Governance framework, data classification
L7: Agent Ecosystem	MCP servers, ClawHub, inter-agent trust	MCP governance, skill registry

AICM control domain alignment maps the Zero Trust reference architecture controls across deployment phases:

AICM Control Domain	ZT Controls	Implementation Layer	Deployment Phase
AC: Access Control	Per-action authorization, JIT credentials	Identity, Policy, Enforcement	Phase 1-2
AU: Audit and Accountability	Operation logging, delegation chain records	Monitoring	Phase 1-3
CM: Configuration Management	Agent config as protect surface, immutable configs	Containment	Phase 1
IA: Identification and Authentication	SPIFFE workload identity, A2A Security Cards	Identity	Phase 1
IR: Incident Response	Anomaly detection, automatic demotion, circuit breakers	Monitoring, Containment	Phase 2-3
MP: Media Protection	Data classification, parameter sensitive data detection	Enforcement	Phase 2

AICM Control Domain	ZT Controls	Implementation Layer	Deployment Phase
PE: Physical and Environmental	Container isolation, resource limits, namespace separation	Containment	Phase 2
PL: Planning	Protect surface definition, transaction flow mapping	Policy	Phase 0
RA: Risk Assessment	Composite autonomy assessment, behavioral risk scoring	Monitoring, Policy	Phase 2-3
SA: System and Services Acquisition	Supply chain verification, code signing, internal registries	Enforcement	Phase 2-3
SC: System and Communications	SDP, SPA, microsegmentation, encrypted channels	Enforcement, Containment	Phase 2
SI: System and Information Integrity	Tool description integrity, model integrity, behavioral bounds	Enforcement, Monitoring	Phase 2-3

22. Regulatory and Compliance Alignment

OpenClaw-based agents operating in regulated domains introduce regulatory obligations that technical controls alone do not satisfy. The EU AI Act establishes requirements for human oversight (Article 14) that Zero Trust's continuous verification, human-in-the-loop gates, and dynamic autonomy adjustment architecturally satisfy. Risk management (Article 9) is operationalized through protect surface identification and risk-based policy frameworks. Data governance (Article 10) is enforced through data access controls and sensitive data detection. Transparency (Article 13) is provided through comprehensive audit logging and delegation chain records. Robustness (Article 15) is achieved through defense-in-depth architecture maintaining security even when individual controls fail. The August 2026 deadline for high-risk AI system compliance requires organizations to assess whether agent use cases fall within Annex III categories and build documentation, conformity assessment, and human oversight mechanisms from Phase 2 onward [52].

Regulation	Agent-Specific Concern	Phase Applicability
SOX (Section 404)	Agents accessing financial systems must produce audit trails satisfying internal control requirements; agent-generated outputs must be attributable and reproducible	Phase 2+
HIPAA	Agents processing PHI require Business Associate Agreements with model providers, minimum necessary access controls, and breach notification accounting for agent-speed data access	Phase 2+

Regulation	Agent-Specific Concern	Phase Applicability
PCI-DSS	Agents interacting with payment card data expand PCI scope; network segmentation must account for MCP communication paths	Phase 2+
CCPA/CPRA	Agent processing of personal data triggers access request and deletion obligations; agent memory and log retention must be governable	Phase 2+
SEC Cyber Disclosure	Material agent compromise at a public company may trigger SEC disclosure obligations; IR must include materiality assessment	Phase 0+

Cyber insurance coverage should be verified to extend to agent-initiated actions and agent-facilitated breaches. Policies are already introducing AI-specific exclusions and sublimits; the maturity framework's controls — particularly execution isolation, ephemeral credentials, and supply chain governance — may be required by insurers as preconditions for coverage.

23. Resource Planning and Organizational Governance

The following estimates provide rough order-of-magnitude guidance for a mid-size enterprise (1,000-5,000 employees, 200-500 developers).

Phase	Duration	FTE (Security)	Infrastructure	Engineering Investment	Annual Run Rate
Phase 0	2-4 weeks	0.5	None	40-80 hrs	N/A (one-time)
Phase 1	1-3 months	0.25-0.5	$500-2K/mo	80-120 hrs setup	$10-30K/yr
Phase 2	3-6 months	1-2	$2-10K/mo	200-400 hrs IAM integration	$150-400K/yr
Phase 3	Ongoing	2-4	$5-20K/mo	Variable	$300K-1M/yr

Organizations should extend their data classification framework to include agent access as a classification dimension:

Data Classification	Agent Access	Required Controls
Public	Permitted	Baseline hardening
Internal	Permitted with controls	Workspace isolation, egress filtering, logging

Data Classification	Agent Access	Required Controls
Confidential	Restricted	NemoClaw or equivalent sandboxing, DLP, approval required
Highly Confidential / Regulated	Prohibited	Agent access blocked; human processing only

For board communication, CISOs should frame the risk concisely: "AI agents that can autonomously access email, files, databases, and cloud services are already running on approximately N percent of our developer workstations as unapproved software. The most widely used agent framework has experienced multiple critical security vulnerabilities in 2026, including a supply chain attack that compromised 20 percent of its extension registry. We have established a four-phase governance program to bring these tools under management." Key metrics for ongoing reporting include discovered versus approved agent instances, current maturity phase by business unit, agent-related security incidents, compliance status for regulated use cases, and program cost against budget.

24. The Agentic Control Plane Imperative

Agentic AI is entering a phase of rapid scaling. NVIDIA's entry into agent orchestration, the maturation of interoperability standards, and the momentum of open-source adoption are accelerating both the capability and the complexity of enterprise agent deployments. Enterprises face a fundamental architectural choice: treat agents as individual applications to be secured in isolation, producing fragmented controls, reactive security posture, and compounding risk — or treat agents as infrastructure requiring its own governance architecture, its own identity model, its own observability plane, and its own security controls designed to scale with the agent population rather than against it.

The Agentic Control Plane is the Zero Trust enforcement layer governing all agent actions through five architectural layers: the Identity Layer providing cryptographic identity for all entities through SPIFFE/SPIRE or cloud IAM; the Policy Layer defining and distributing policies expressed in OPA Rego or Cedar; the Enforcement Layer interposing policy enforcement between agents and all resources; the Monitoring Layer providing continuous behavioral verification; and the Containment Layer bounding blast radius through process isolation, network microsegmentation, credential scoping, and delegation scope narrowing. The control plane's decisions are centralized and consistent while enforcement is distributed across every point where an agent interacts with a resource, another agent, or the external world.

The durable requirement is not any single product, framework, or protocol. It is the architectural discipline to govern agents as infrastructure — ensuring that as agent capabilities scale, governance, assurance, and observability scale with them. The organizations that succeed in the agentic era will not be those with the most advanced agents. They will be those that build and operate a durable agentic control plane.

25. Architect Quick Guide

Top 5 Risks: credential theft and standing access (demonstrated by Salesloft Drift's 700+ organization exposure through a single standing token); supply chain poisoning (ClawHavoc's 800+ malicious skills on ClawHub); prompt injection as C2 (Agent Commander's full command-and-control framework); autonomy escalation (Alibaba ROME's autonomous cryptocurrency mining); and delegation chain privilege escalation (Islands of Agents research demonstrating composite authority accumulation).

Top 5 Controls: JIT credentials replacing all standing credentials with short-lived per-task tokens (5-15 minute TTL); MCP proxy enforcement layer routing all agent-to-tool traffic through policy evaluation; internal skill registry with code signing, behavioral analysis, and provenance verification; network isolation separating MCP, A2A, and AG-UI traffic with blocked unapproved outbound connections; and behavioral monitoring with dynamic autonomy adjustment automatically reducing autonomy when anomalies are detected.

Minimum Viable Control	What It Prevents	Implementation Options
Just-in-Time Credentials	Standing credential theft, cross-tenant exposure, persistent blast radius	HashiCorp Vault, AWS STS, Azure managed identity, GCP workload identity federation
MCP Proxy / Enforcement Layer	Direct agent-to-tool exploitation, unauthorized tool invocations, parameter injection	Sidecar proxy (Envoy + OPA), API gateway with policy plugin, dedicated MCP gateway
Supply Chain Control	Malicious skill installation, dependency poisoning, tool description manipulation	Internal skill registry, code signing with transparency log, behavioral analysis pipeline
Network Isolation	Data exfiltration, lateral movement, C2 communication, WebSocket hijacking	Kubernetes network policies, VPC security groups, SDP/SPA, host-based firewall rules
Comprehensive Logging	Undetected compromise, investigation inability, behavioral monitoring blind spots	Agent telemetry to SIEM, structured tool call logging, immutable audit log

Key Architecture Decisions: Sidecar vs. gateway for MCP enforcement (sidecar provides isolation without single points of failure but increases complexity; gateway simplifies operations but requires HA engineering). Identity system selection (SPIFFE/SPIRE for multi-cloud/hybrid; cloud-native IAM for single-cloud). Policy language (OPA Rego for complex conditional logic; Cedar for readability and AWS integration). Monitoring architecture (OpenTelemetry to existing SIEM for organizations with investment; dedicated agent monitoring for greenfield).

26. Quick-Start Hardening Checklist

Priority	Control	Complexity	Primary Threat Addressed	AICM Domain
Critical	Bind gateway to 127.0.0.1, never 0.0.0.0	Low	Internet exposure (135K instances)	Infrastructure Security
Critical	Enforce Gateway auth (32-char token minimum)	Low	ClawJacked WebSocket hijack	Application Security
Critical	Update to v2026.2.26+	Low	All pre-March CVEs	Change Control
Critical	Rotate all standing API keys and credentials	Medium	Credential theft / Salesloft Drift pattern	Identity and Access Mgmt
High	Disable auto-approval of tool invocations	Low	Prompt injection escalation to RCE	Application Security
High	Restrict working directory to designated workspace	Low	Unauthorized file access upon compromise	Data Security
High	Block ClawHub direct installation; use internal registry	Medium	ClawHavoc supply chain	Supply Chain Mgmt
High	Restrict MCP connections to allowlisted servers	Medium	Tool poisoning / rug pull	Application Security
High	Lock SOUL.md and HEARTBEAT.md to read-only	Low	Behavioral persistence attacks	Configuration Mgmt
High	Deploy file integrity monitoring on config directory	Medium	Silent configuration modification	Logging and Monitoring
Medium	Implement Docker hardening (cap-drop, read-only, seccomp)	Medium	Sandbox escape (CVE-2026-24763)	Infrastructure Security
Medium	Deploy NemoClaw OpenShell (if NVIDIA hardware available)	Medium	All runtime exploitation vectors	Infrastructure Security
Medium	Configure network egress allowlisting	Medium	Data exfiltration via prompt injection	Infrastructure Security
Medium	Enable comprehensive tool invocation logging to SIEM	Medium	Undetected compromise and behavioral drift	Logging and Monitoring

Priority	Control	Complexity	Primary Threat Addressed	AICM Domain
Medium	Implement JIT credential provisioning	High	Standing credential exposure	Identity and Access Mgmt

CSA Resource Alignment

This paper draws upon and integrates guidance from multiple CSA frameworks. MAESTRO provides the seven-layer threat modeling methodology applied throughout the security landscape analysis and control mapping. The AI Controls Matrix (AICM) v1.0 provides the control objective taxonomy used in framework alignment, and organizations building compliance programs should reference its Governance, Risk, and Compliance domain for organizational controls that technical stacks do not address. The Cloud Controls Matrix (CCM) provides threat and vulnerability management controls defining patch SLA frameworks applicable to OpenClaw version management. CSA's Agentic AI Identity and Access Management guidance provides the foundational identity architecture recommendations including DID and Verifiable Credential primitives for portable agent identity. The Zero Trust Advancement Center (ZTAC) provides architectural principles for per-action authorization. STAR for AI extends registry and certification frameworks to AI systems, providing mechanisms for validating agent deployment security posture. Valid-AI-ted provides complementary automated validation tooling. The AI Risk Observatory provides ongoing intelligence on emerging agentic AI threats. CSA's role as a CVE Numbering Authority (CNA) for AI security vulnerabilities ensures vulnerability disclosure infrastructure keeps pace with ecosystem growth. The Agentic AI Red Teaming Guide provides testing methodology for validating control effectiveness. The Islands of Agents framework provides agent categorization for identity architecture decisions.

References

[1] CNBC. "From Clawdbot to Moltbot to OpenClaw: Meet the AI Agent Generating Buzz and Fear Globally." February 2, 2026.

[2] TechCrunch. "OpenClaw Creator Peter Steinberger Joins OpenAI." February 15, 2026.

[3] Palo Alto Networks. "Why Moltbot May Signal an AI Security Crisis." January 2026.

[4] Adversa AI. "OpenClaw Security 101: Vulnerabilities and Hardening." 2026.

[5] The Hacker News. "OpenClaw Bug Enables One-Click Remote Code Execution via Malicious Link." February 2026.

[6] SOCRadar. "CVE-2026-25253: 1-Click RCE in OpenClaw Through Auth Token Exfiltration." February 2026.

[7] GitLab Advisory Database. "GHSA-g27f-9qjv-22pm: OpenClaw Log Poisoning." February 2026.

[8] The Hacker News. "ClawJacked Flaw Lets Malicious Sites Hijack Local OpenClaw AI Agents via WebSocket." February 2026.

[9] Oasis Security. "ClawJacked: OpenClaw Vulnerability Enables Full Agent Takeover." February 25, 2026.

[10] Endor Labs. "CVE-2026-24763: Docker Sandbox Escape Through PATH Manipulation in OpenClaw." February 2026.

[11] The Hacker News. "Researchers Find 341 Malicious ClawHub Skills Stealing Data from OpenClaw Users." February 2026.

[12] Snyk Security Research. "ClawHub Marketplace Analysis: 1,467 Malicious Entries, 91% Combining Prompt Injection with Malware." February 2026.

[13] Cisco Security Blog. "I Run OpenClaw at Home. That's Exactly Why We Built DefenseClaw." March 2026.

[14] Adversa AI. "OpenClaw Security 101." 2026 (citing Moltbook database exposure).

[15] B. Van Roo. "The Agent Internet Just Went Live." Substack, February 2026.

[16] TechCrunch. "Meta Acquired Moltbook, the AI Agent Social Network." March 10, 2026.

[17] SecurityScorecard STRIKE Team. "135K OpenClaw Instances Exposed: Internet-Wide Scanning Results." February 2026.

[18] WZ-IT. "OpenClaw Secure Deployment Security Hardening Guide." 2026 (citing JFrog research).

[19] Dark Reading. "OpenClaw AI Runs Wild in Business Environments." February 2026 (citing Token Security).

[20] IBM Security. "Cost of a Data Breach Report 2025." IBM, 2025.

[21] Penligent AI. "SOUL.md Persistence Attacks: Creating Durable C2 Channels Through OpenClaw's Context Memory System." February 2026.

[22] Penligent AI. "Soul Packs: Steganographic Injection in OpenClaw Identity Templates." March 2026.

[23] HiddenLayer. "HEARTBEAT.md Command-and-Control Implant Demonstration." March 2026.

[24] Heyuan. "MCP Security 2026: 30 CVEs in 60 Days." March 10, 2026.

[25] BlueRock Security. "MCP Server Security Analysis of 7,000+ Servers." 2026.

[26] OWASP Agentic Security Initiative. "OWASP Top 10 for Agentic Applications 2026." OWASP Foundation, 2026.

[27] Oligo Security. "Critical RCE Vulnerability in Anthropic MCP Inspector (CVE-2025-49596)." 2025.

[28] GitHub Advisory Database. "CVE-2025-66416: MCP Python SDK – No DNS Rebinding Protection by Default." 2025.

[29] Practical DevSecOps. "MCP Security Vulnerabilities." 2026.

[30] NVIDIA Newsroom. "NVIDIA Announces NemoClaw for the OpenClaw Community." March 16, 2026.

[31] VentureBeat. "Nvidia Lets Its 'Claws' Out: NemoClaw Brings Security, Scale to the Agent Platform Taking Over AI." March 2026.

[32] Repello AI. "What Is NVIDIA NeMoClaw? A Security Engineer's First Look." March 2026.

[33] Futurum Group. "At GTC 2026, NVIDIA Stakes Its Claim on Autonomous Agent Infrastructure." March 2026.

[34] Anthropic. "Claude Cowork: Research Preview." 2026.

[35] OpenAI. "Introducing Codex." 2025.

[36] Model Context Protocol. "Authorization Specification (Draft)." modelcontextprotocol.io, 2026.

[37] Google Cloud Blog. "Agent2Agent Protocol Is Getting an Upgrade." 2026.

[38] A2A Protocol. "A2A Protocol Specification." a2a-protocol.org, 2026.

[39] CopilotKit. "AG-UI Protocol Overview." AG-UI Documentation, 2026.

[40] AWS. "Amazon Bedrock AgentCore Runtime Adds AG-UI Protocol Support." March 13, 2026.

[41] GetStream. "Top AI Agent Protocols 2026." GetStream Blog, 2026.

[42] Cloud Security Alliance. "Zero Trust Guiding Principles v1.1" and "Software-Defined Perimeter Specification." ZTAC Working Group.

[43] Cloud Security Alliance. "AI Security: IAM Delivered at Agent Velocity." February 17, 2026.

[44] Arize AI. "100 AI Agents Per Employee: The Enterprise Governance Gap." 2026.

[45] R. Mogull (CSA). "Islands of Agents: Why One IAM to Rule Them All Doesn't Work." March 10, 2026.

[46] NIST NCCoE. "Accelerating the Adoption of Software and AI Agent Identity and Authorization." February 5, 2026.

[47] Cloud Security Alliance AI Safety Initiative. "Agentic AI Identity and Access Management: A New Approach." August 2025.

[48] Cloud Security Alliance. "Zero Trust for Large Language Models." ZTAC Working Group, 2025.

[49] Cloud Security Alliance. "AI Security: When Your Agent Crosses Multiple Independent Systems, Who Vouches for It?" March 11, 2026.

[50] Knostic. "openclaw-telemetry: Agent Monitoring with JSONL and SIEM Integration." 2026.

[51] MITRE ATLAS. "OpenClaw Case Studies CS0048-CS0051: Supply Chain, RCE, CSRF, and Prompt Injection Attack Chains." February 2026.

[52] European Parliament. "Regulation (EU) 2024/1689 – Artificial Intelligence Act." Official Journal of the European Union, 2024.

[53] Cloud Security Alliance. "MAESTRO: Multi-Agent Environment, Security, Threat, Risk, and Outcome Framework." February 2025.

[54] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.0." July 2025.

[55] Cloud Security Alliance. "AI Security: When Authorization Outlives Intent." February 25, 2026.

[56] MintMCP Blog. "Every OpenClaw CVE Explained: What Enterprise Security Teams Must Patch 2026." 2026.

[57] Kaspersky. "OpenClaw Security Audit: 512 Vulnerabilities Identified." 2026.

[58] SecurityScorecard STRIKE Team. "How Exposed OpenClaw Deployments Turn Agentic AI Into an Attack Surface." February 2026.

[59] Northflank. "How to Sandbox AI Agents in 2026: MicroVMs, gVisor & Isolation Strategies." 2026.

[60] Noma Security. "Can AI Agents Go Rogue? The Risk of Goal Misalignment." 2026.

Appendix A: OpenClaw Security Audit Tool

The Cloud Security Alliance has released an open-source companion tool that automates compliance scanning against the controls defined in this whitepaper. The **OpenClaw Security Audit Tool** is available at github.com/CloudSecurityAlliance/openclaw-audit and is designed to help security teams rapidly assess OpenClaw installations against the hardening guidance presented throughout this document.

Overview

The OpenClaw Security Audit Tool is a static analysis scanner that evaluates OpenClaw installations and deployment repositories against the controls established in this whitepaper. It produces structured security gap reports with optional auto-remediation capabilities, requiring zero external dependencies beyond the Python 3.10+ standard library.

Features

The tool provides **53 security checks** organized across nine categories: Configuration, SOUL.md Integrity, Skill Vetting, MCP Server, File Permissions, Credentials, Docker Sandbox, Network Exposure, and Version. It auto-detects installed instances (`~/.openclaw/`), git repositories (Docker Compose, NemoClaw), and hybrid environments. Remote SSH scanning is supported for individual hosts or entire fleets via the `--hosts` option, and the tool generates output in four formats: terminal (colored), JSON, SARIF 2.1.0 (for GitHub Advanced Security integration), and Markdown.

An auto-fix engine operates at three levels of aggressiveness. The *basic* level corrects file permissions, including setting SOUL.md to read-only, configuration files to mode 600, and directories to mode 700. The *medium* level adds configuration hardening such as authentication enforcement, loopback binding, mDNS disabling, execution denial, tool deny lists, SSRF prevention, and workspace-only restrictions. The *complete* level adds quarantining of suspicious skills, sandbox mode enforcement, elevated tool disabling, and Docker hardening. All fixes create `.bak` backups before modifying configuration files, and fixes applied to remote hosts are automatically pushed back over SSH.

Every finding is tagged with mappings to the OWASP Top 10 for Agentic Applications (ASI01–ASI10), MITRE ATLAS agent techniques (AML.T0040–AML.T0105), MAESTRO agentic AI security layers (L1–L7), CSA AI Controls Matrix (AICM) domains, and the specific whitepaper section that defines the corresponding control.

Quick Start

```
# Install
pip install -e .

# Scan an OpenClaw installation
openclaw-audit ~/.openclaw/

# Scan a deployment repo
openclaw-audit ./my-openclaw-repo/

# Or run directly
python3 -m openclaw_audit ~/.openclaw/
```

Usage

```
openclaw-audit [-h] [-f {terminal,json,sarif,markdown,all}] [-o OUTPUT]
               [--fix] [--fix-level {basic,medium,complete}]
               [--dry-run] [--quiet]
               [--hosts HOSTS_FILE] [--ssh-key KEY] [--password]
               [target]
```

Arguments and Options

Argument / Flag	Description
`target`	Local path or remote target (`user@host:/path`). Omit if using `--hosts` .
`-f` , `--format`	Output format: `terminal` (default), `json` , `sarif` , `markdown` , `all`
`-o` , `--output`	Output file path. With `--hosts` , this is a directory for per-host reports.
`--fix`	Apply auto-fixes. Fixes are pushed back to remote hosts via SSH.
`--fix-level`	Fix aggressiveness: `basic` , `medium` , `complete` (default)
`--dry-run`	Show what fixes would be applied without making changes
`-q` , `--quiet`	Suppress terminal output

Argument / Flag	Description
`--hosts`	File with one target per line for fleet scanning
`--ssh-key`	Path to SSH private key for remote connections
`-p`, `--password`	Prompt for SSH password. With `--hosts`, prompts once and reuses for all hosts.

Examples

```
# Terminal report (default)
openclaw-audit ~/.openclaw/

# JSON output to file
openclaw-audit ~/.openclaw/ --format json -o report.json

# SARIF for CI/CD (GitHub Advanced Security)
openclaw-audit ./repo/ --format sarif -o results.sarif

# Markdown report with framework coverage tables
openclaw-audit ~/.openclaw/ --format markdown -o report.md

# All formats at once
openclaw-audit ~/.openclaw/ --format all -o audit-report

# Preview what fixes would be applied
openclaw-audit ~/.openclaw/ --fix --dry-run

# Apply medium-level fixes (config hardening)
openclaw-audit ~/.openclaw/ --fix --fix-level medium

# Apply all fixes (includes skill quarantine + Docker hardening)
openclaw-audit ~/.openclaw/ --fix --fix-level complete
```

Remote SSH Scanning

The tool supports scanning remote OpenClaw installations over SSH without requiring installation on the target host. Files are fetched via SSH+tar, scanned locally, and fixes (if applied) are pushed back automatically.

```
# Scan a remote host (SSH key from agent or ~/.ssh/)
openclaw-audit root@server.example.com:/home/openclaw/

# Specify SSH key
openclaw-audit deploy@10.0.1.50:/opt/openclaw --ssh-key ~/.ssh/deploy_key

# Use password authentication
openclaw-audit admin@host.internal:/home/openclaw -p

# Remote scan with auto-fix (fixes pushed back via SSH)
openclaw-audit root@server:/home/openclaw --fix --fix-level medium
```

Fleet Mode

Multiple hosts can be scanned from a file, with each line specifying a target in `user@host:/path` format (lines starting with `#` are ignored). Hostnames without a path default to `root@host:/home/openclaw`.

```
# hosts.txt
root@prod-1.example.com:/home/openclaw
root@prod-2.example.com:/home/openclaw
deploy@staging:/opt/openclaw

# Scan all hosts, save per-host reports to a directory
openclaw-audit --hosts hosts.txt --format markdown -o reports/

# Fleet scan with password auth (prompts once, reuses for all)
openclaw-audit --hosts hosts.txt -p

# Fleet scan with fixes
openclaw-audit --hosts hosts.txt --fix --dry-run
```

Check Categories

Category	Checks	What It Scans
Configuration	18	Gateway auth, binding, mDNS, shell exec, sandbox, SSRF, tool groups, auto-update, Moltbook heartbeat, token strength, config.patch, sensitive dirs

Category	Checks	What It Scans
SOUL.md Integrity	7	Base64 payloads, zero-width Unicode, shell commands, override patterns, C2 URLs
Skill Vetting	6	Prompt injection, ClawHavoc patterns, obfuscation, credential access, exfiltration
MCP Server	5	Authentication, version pinning, TLS, tool description poisoning, public source detection
File Permissions	2	Config directory (700), credential files (600)
Credential Hygiene	4	API keys in configs, .env files, session transcripts, standing long-lived keys
Docker Sandbox	7	cap_drop, read-only root, seccomp, dangerous mounts, privileged mode
Network Exposure	2	Docker host network mode, egress restrictions
Version	2	Minimum safe version (v2026.2.26+)

Fix Levels

Level	What It Does
`basic`	File permissions (SOUL.md read-only, config 600, directory 700)
`medium`	All basic + config hardening (auth, loopback binding, mDNS off, exec deny, tool deny lists, SSRF off, workspace-only)
`complete`	All medium + quarantine suspicious skills, sandbox mode, elevated tools off, Docker hardening

Framework Mappings

Every finding includes mappings to the OWASP Top 10 for Agentic Applications (ASI01–ASI10), MITRE ATLAS agent techniques (AML.T0040–AML.T0105), MAESTRO agentic AI security layers (L1–L7), CSA AI Controls Matrix (AICM) domains, and the corresponding whitepaper section reference. The Markdown report format includes coverage summary tables for OWASP Agentic Top 10, MAESTRO layers, and Phase 0 emergency checklist alignment.

CI/CD Integration

The tool integrates with GitHub Actions via SARIF output for GitHub Advanced Security:

```
- name: OpenClaw Security Audit
  run: |
    pip install openclaw-audit
    openclaw-audit ./openclaw-config/ --format sarif -o results.sarif

- name: Upload SARIF
  uses: github/codeql-action/upload-sarif@v3
  with:
    sarif_file: results.sarif
```

Exit Codes

The tool returns exit code `0` when all checks pass (or only warnings are present) and exit code `1` when one or more checks fail.

License

The OpenClaw Security Audit Tool is released under the Apache 2.0 license.