Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

The Rise of Autonomous AI Agents: A Comprehensive Survey of OpenClaw —Architecture, Security, Ecosystem, and Beyond

Siyuan Li

liaoyanqing666@gmail.com

University of Georgia https://orcid.org/0009-0007-7094-3193

Research Article

Keywords: autonomous AI agents, OpenClaw, agent security, ASTELD taxonomy, AI agent frameworks

Posted Date: August 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10596296/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

The Rise of Autonomous AI Agents: A Comprehensive Survey of OpenClaw — Architecture, Security, Ecosystem, and Beyond

Siyuan Li[1], Peng Shu[1], Churan Yu[2], Peilong Wang[3], Ruidong Zhang[1], Bowen Guo[4], Xinliang Li[1], Ruiyu Yan[5], Arif Hassan Zidan[6], Yi Pan[1], Wei Ruan[1], Lifeng Chen[1], Junhao Chen[1], Zhaojun Ding[1], Yiwei Li[1], Zhengliang Liu[1], Haixing Dai[7], Lin Zhao[8], Yu Bao[4], Xiang Li[9], Wei Zhang[6], and Tianming Liu*[1]

[1]School of Computing, University of Georgia, Athens, GA, USA
[2]College of Engineering, University of Georgia, Athens, GA, USA
[3]Department of Radiation Oncology, City of Hope National Medical Center, Duarte, CA, USA
[4]Department of Graduate Psychology, James Madison University, VA, USA
[5]Tandon School of Engineering, New York University, New York, NY, USA
[6]School of Computer and Cyber Sciences, Augusta University, Augusta, GA, USA
[7]Meta, Menlo Park, CA, USA
[8]Department of Biomedical Engineering, New Jersey Institute of Technology, Newark, NJ, USA
[9]Department of Radiology, Massachusetts General Hospital and Harvard Medical School, Boston, MA, USA

Abstract

Autonomous AI agents use large language models as controllers that plan, invoke tools, and act on the world, marking a shift from text generation to autonomous action. No project captures both the promise and the peril of this transition more clearly than OpenClaw. Released in November 2025, it accumulated 360,000+ GitHub stars by late April 2026, surpassing both React and the Linux kernel. Over the same period, public trackers recorded 100+ advisories and 10+ published CVEs; broader vendor summaries reported still larger totals under different aggregation rules; and the coordinated ClawHavoc supply chain attack exposed 1,184 malicious marketplace skills. This paper presents the first comprehensive, multi-dimensional survey of OpenClaw and the autonomous AI agent landscape, treating OpenClaw as a sociotechnical system across five dimensions—architecture, security, ecosystem comparison, derivative ecosystem, and adoption dynamics—with governance as a cross-cutting theme. We contribute (1) the most extensive multi-dimensional survey of OpenClaw to date; (2) a six-category security vulnerability taxonomy with cross-source triangulation from five institutional assessments; (3) the ASTELD classification framework—a six-axis taxonomy (Architecture, Security, Tool integration, Execution, Level of autonomy, Deployment) validated against eight frameworks; (4) the first systematic mapping of 50+ derivative projects across three tiers; and (5) eight formulated research questions spanning architectural, security, ecosystem, and sociotechnical domains. Our central thesis is that OpenClaw's trajectory—from weekend project to the most-starred repository in GitHub history to a case study in security debt—reveals structural tensions inherent to the autonomous AI agent paradigm. Three such tensions recur throughout this analysis: the accessibility–security trade-off, the open-source speed paradox, and the impossibility of a single architecture satisfying all use cases.

Keywords: autonomous AI agents; OpenClaw; agent security; ASTELD taxonomy; AI agent frameworks

*Correspondence: tliu@uga.edu

Contents

1 Introduction **4**

2 Background: From Chatbots to Autonomous AI Agents **5**
2.1 LLM-Based Applications 5
2.2 Definition of Autonomous AI Agents 6
2.3 The Security Implications of the Agent Paradigm 6

3 OpenClaw: Origins and Evolution **7**
3.1 From Clawdbot to OpenClaw 7
3.2 Growth Trajectory and Milestones 7
3.3 Governance Transitions 8
3.4 Naming and Identity 9
3.5 Summary 9

4 System Architecture **10**
4.1 Architectural Overview 11
4.2 Execution Model 11
4.3 The Lethal Trifecta 12
4.4 State and Memory Management 12
4.5 Deployment Topology 13
4.6 Architectural Comparison via ASTELD 13
4.7 Summary 14

5 Security and Privacy Analysis **14**
5.1 Vulnerability Taxonomy 14
5.1.1 Category A: Authentication and Session Trust Abuse 14
5.1.2 Category B: Approval System Bypass 15
5.1.3 Category C: Operating System-Specific Escapes 16
5.1.4 Category D: Prompt Injection 16
5.1.5 Category E: Supply Chain Attacks (ClawHavoc) 17
5.1.6 Category F: Infrastructure Misconfigurations 17
5.2 Attack Chain Taxonomy 18
5.3 Institutional Security Assessments 18
5.4 Defense Landscape 19
5.5 Summary 19

6 Ecosystem Comparison and the ASTELD Classification Framework **20**
6.1 Motivation and Scope 20
6.2 The ASTELD Classification Framework 20
6.3 Framework Mapping 21
6.4 Cross-Cutting Patterns 23
6.5 Comparative Analysis by Dimension 24
6.6 Framework Selection Matrix 25
6.7 Summary 25

7 Adoption, Social Impact, and the Derivative Ecosystem **25**
7.1 Adoption Metrics and Growth Dynamics 26
7.1.1 GitHub Star Growth 26
7.1.2 Comparative Platform Metrics 26
7.1.3 Usage and Ecosystem Metrics 26
7.1.4 The Adoption Paradox: Stars ≠ Enterprise Adoption 27
7.2 The Derivative Ecosystem 27
7.2.1 Derivative Taxonomy 27
7.2.2 Derivative Prediction Power 28
7.3 Social and Regulatory Impact 29
7.3.1 The Token Economy 29
7.3.2 ClawHub Marketplace Dynamics 29
7.3.3 Regulatory Responses 30
7.3.4 Enterprise Adoption Dynamics 30
7.4 Summary 30

8 Open Research Questions and Future Directions **31**
8.1 Architectural Research 31
8.2 Security Research 32
8.3 Ecosystem Research 32
8.4 Sociotechnical Research 33
8.5 Meta-Observations 34

9 Conclusion **34**

1 Introduction

The emergence of autonomous artificial intelligence (AI) agents represents a paradigmatic shift in how humans interact with AI. Unlike chatbots that generate text in response to prompts, autonomous agents decompose goals into subtasks, invoke tools, access file systems, and communicate with external services—all with minimal human oversight. This transition from *generation* to *action* has profound implications for software engineering, security, and governance.

These implications are no longer theoretical. In the span of a few months, a single open-source project, OpenClaw, has made them concrete. Released as a weekend side project in November 2025 under the name *Clawdbot*, OpenClaw accumulated 60,000 GitHub stars within four days of going viral in late January 2026, surpassed React as the most-starred software project on GitHub by March 3, 2026 with 250,829 stars, and reached 360,000+ stars on GitHub by late April 2026 [1][2][3][4]. In parallel, public trackers documented 100+ tracked advisories and 10+ published CVEs by April 2026, while broader vendor summaries reported larger tallies of tracked vulnerability records under different aggregation rules [5][6]; OpenClaw also suffered a coordinated supply chain attack, *ClawHavoc*, that ultimately led to the identification of 1,184 malicious skills in its marketplace [7][8]. This is the paradox: the most-starred repository on GitHub is also a case study in security debt. The tensions it reveals are not unique to one project; they are built into the agent paradigm itself.

Three such tensions recur throughout this analysis. First, the *accessibility–security trade-off*: the design choices that made OpenClaw accessible to non-technical users (a messaging-platform interface, a monolithic daemon with system-level access, a permissionless skill marketplace) are precisely the choices that created its most serious vulnerabilities. Second, the *open-source speed paradox*: rapid community-driven development outpaced the project's capacity to establish security governance, resulting in a 4-day burst in which 9 new security-tracker entries were logged between March 18 and March 21, 2026 [5]. Third, the *impossibility of architectural universality*: no single agent architecture can simultaneously optimize for accessibility, security, expressiveness, and enterprise readiness—a finding we demonstrate through systematic cross-framework comparison.

Despite the intensity of public attention the OpenClaw project has received, the academic literature has not addressed these tensions as a whole. Existing work covers individual aspects—security taxonomies [9][10], ecosystem surveys [11], empirical studies of agent-to-agent interaction on Moltbook [12][13], and critical audits of the hype cycle [14]—but no prior work provides a comprehensive, multi-dimensional survey that simultaneously examines architecture, security, ecosystem position, derivative fragmentation, adoption dynamics, and governance.

To fill this gap, this paper presents the first comprehensive survey to treat OpenClaw as a complete sociotechnical system. Specifically, we make five contributions:

1. **First multi-dimensional survey.** The most extensive coverage of OpenClaw to date, spanning five dedicated analytical dimensions—architecture (Section 4), security (Section 5), ecosystem comparison (Section 6), derivative ecosystem (Section 7), and adoption dynamics (Section 7)—with governance examined as a cross-cutting theme across origin history (Section 3), regulatory impact (Section 7), and open research questions (Section 8).

2. **Security vulnerability taxonomy with cross-source triangulation.** A six-

category vulnerability taxonomy synthesized from CVE databases, five institutional security assessments (Microsoft [15], CertiK [16], CrowdStrike [17], Trend Micro [18], and Koi Security [8]), and academic research—including the first comprehensive account of the ClawHavoc supply chain campaign.

3. **ASTELD classification framework.** A six-axis taxonomy (**A**rchitecture, **S**ecurity, **T**ool integration, **E**xecution, **L**evel of autonomy, **D**eployment) for classifying autonomous AI agent platforms, validated against eight frameworks. We show that the ASTELD axis constraints of a base platform correlate with the directions of observed derivative ecosystem fragmentation, providing explanatory power over ecosystem evolution.

4. **Derivative ecosystem mapping.** The first systematic classification of the OpenClaw derivative ecosystem, cataloguing 50+ projects and providing detailed analysis of representative derivatives across three tiers (community forks, enterprise adaptations, research platforms), with quantitative metrics for the top four derivatives drawn from primary GitHub data [19].

5. **Actionable research agenda.** Eight formulated research questions (RQ1–RQ8) spanning architectural, security, ecosystem, and sociotechnical dimensions, each grounded in identified evidence gaps.

The remainder of this paper is organized as follows. Section 2 establishes the intellectual context by tracing the evolution from chatbots to autonomous agents. Section 3 introduces OpenClaw's origins, evolution, and governance transitions. Section 4 provides a technical deep dive into its architecture. Section 5 presents our security and privacy analysis, including the six-category vulnerability taxonomy and attack chain composition. Section 6 positions OpenClaw within the broader AI agent ecosystem through the ASTELD classification framework. Section 7 examines adoption dynamics, social impact, and the derivative ecosystem. Section 8 formulates open research questions and future directions. Finally, Section 9 concludes the work.

2 Background: From Chatbots to Autonomous AI Agents

2.1 LLM-Based Applications

The development of LLM-based applications can be understood as a progression from standalone text generation, to retrieval- and workflow-enhanced systems, and more recently to autonomous agentic architectures.

Text Generation Chatbots. The release of GPT-3 [20] established the paradigm of prompt-in, text-out interaction. Users provided natural language instructions; models returned generated text. Applications included summarization, translation, and question answering. The model had no memory, no tool access, and no ability to take actions in the world.

Retrieval-Augmented Systems. Retrieval-Augmented Generation (RAG) extended LLMs by grounding their outputs in external knowledge bases. Frameworks such as LangChain (released October 2022) [21] provided composable abstractions for chaining

LLM calls with document retrieval, enabling applications like domain-specific chatbots and enterprise knowledge assistants. The model could now reference external data, but still could not act on the world.

Autonomous AI Agents. The agent paradigm represents a qualitative shift: LLMs are no longer endpoints but controllers. An autonomous agent observes its environment, formulates plans, invokes tools (APIs, shell commands, file operations), and iterates until a goal is achieved. AutoGPT, released in March 2023 [22], was the first viral demonstration of this paradigm. The subsequent emergence of multi-agent frameworks—AutoGen [23] for multi-agent conversation, CrewAI [24] for role-based collaboration, LangGraph [21] for stateful graph execution—established autonomous agents as a recognized architectural pattern.

OpenClaw is significant as it represents a further step in the operationalization of autonomous agents. By combining an LLM controller with system-level OS access, a messaging-platform UI (WhatsApp, Telegram, Discord), and an open plugin marketplace, OpenClaw made autonomous AI agents accessible to non-technical users for the first time [25][26]. This democratization is simultaneously its greatest achievement and its most serious liability.

2.2 Definition of Autonomous AI Agents

We adopt the following operational definition, drawing on recent surveys of agentic AI and autonomous agents [27][28][29][30]:

> An **autonomous AI agent** is a software system in which a large language model serves as a cognitive controller that (1) maintains persistent state across interactions, (2) formulates multi-step plans to achieve user-specified goals, (3) invokes external tools and services to execute those plans, and (4) operates with a degree of autonomy that permits action without per-step human approval.

This definition distinguishes agents from chatbots (which lack tool invocation and persistent state), from RAG systems (which lack planning and action), and from traditional automations (which are not LLM or AI-based). The Interface EU classification [31] further stratifies autonomy into five levels, from L1 (tool mode, every action requires instruction) through L5 (full autonomous operation, no human oversight). As we show in Section 4, OpenClaw's design enables autonomous operations up to L4 (agent decomposes and executes complex goals end-to-end) [15].

2.3 The Security Implications of the Agent Paradigm

The transition from chatbots to agents introduces a fundamentally broader security surface. A useful framing, sometimes described as the "lethal trifecta," is the combination of (1) access to private data on the user's device, (2) exposure to untrusted external content from external sources, and (3) the ability to communicate with or act upon external services. When all three conditions hold, prompt injection becomes not merely a content moderation problem but an authorization bypass—a malicious instruction embedded in retrieved content can cause the agent to exfiltrate data, install software, or modify system configurations [15][32][33].

This threat model is not hypothetical. Public reporting on OpenClaw in early 2026 documented a rapidly growing stream of vulnerabilities and abuse cases, including CVE-2026-25253 (ClawJacked: cross-site WebSocket hijacking, CVSS 8.8) [34] and CVE-2026-32922 (privilege escalation) [35]. The ClawHavoc campaign demonstrated that supply chain attacks against agent plugin marketplaces can achieve unprecedented scale: Koi Security's initial audit identified 341 malicious skills among 2,857 on ClawHub (approximately 12%), and subsequent reporting later placed the total at 1,184 as the investigation widened [7][8]. These incidents are detailed in Section 5.

OpenClaw emerged at the inflection point of this transition—and its trajectory reveals both the promise and the peril of the agent paradigm.

3 OpenClaw: Origins and Evolution

3.1 From Clawdbot to OpenClaw

OpenClaw began its life as **Clawdbot**, a weekend side project created by Austrian developer Peter Steinberger in November 2025 [36]. Steinberger, previously known as the founder of PSPDFKit (a cross-platform PDF SDK company), built Clawdbot as a personal experiment: a daemon process that connected a large language model to the local operating system via messaging platforms—WhatsApp, Telegram, and later Discord and Slack. The original vision was deceptively simple: enable users to interact with their computers through natural language messages sent from their phones [37][38].

In its earliest form, Clawdbot ran as a local process on macOS, accepting instructions via WhatsApp and executing them through shell commands, file operations, and API calls. There was no plugin system, no marketplace, and no multi-user support. The project attracted modest attention in its first two months—respectable but unremarkable by open-source standards [1].

The transformation from niche tool to global phenomenon began on January 29, 2026, when the project was rebranded to **OpenClaw** after a brief interim rename to **Moltbot** on January 27 following trademark concerns raised by Anthropic. The OpenClaw rebrand coincided with a restructured architecture that included a skill system, a nascent extension marketplace (ClawHub), and support for multiple LLM backends [1]. The timing coincided with a wave of social media attention: within 48 hours, OpenClaw was accumulating stars at a rate of 710 per hour—17,084 per day—a velocity without precedent in GitHub's history [1][2]. By February 2, 2026, within four days of the rebrand, the project had accumulated over 60,000 new stars [1].

3.2 Growth Trajectory and Milestones

The growth of OpenClaw can be divided into four phases, each with distinct dynamics (see Figure 3):

Phase 1: Viral Ignition (January 29 – February 2, 2026). The initial burst was driven by technology media coverage and social media amplification. The project's appeal was immediate: unlike existing AI agent frameworks that required programming knowledge, OpenClaw could be installed by non-technical users and accessed through familiar messaging interfaces. This four-day window produced 60,000 stars at an average of 15,000 per day [1][2].

Phase 2: Sustained Acceleration (February 2 – March 3, 2026). The viral wave did not follow the typical pattern of rapid decay. Instead, growth sustained at approximately 6,600 stars per day, driven by tutorial content, derivative projects, and enterprise experimentation. By February 24, OpenClaw had crossed 224,000 stars and surpassed the Linux kernel (218,000 stars accumulated over 30 years), becoming one of the most-starred repositories in GitHub history [2][39]. The velocity comparison is instructive: Linux accumulated 100,000 stars in approximately 30 years (20 per day average); React reached 100,000 in approximately 10 years (68 per day); OpenClaw reached 100,000 in 12 days—a velocity 123 times that of React and 419 times that of Linux [19][39].

Phase 3: China Wave (March 3 – March 24, 2026). After a brief deceleration, OpenClaw experienced a pronounced March reacceleration driven by adoption in China. Traffic from Chinese IP addresses surged by 1,436% month-over-month, propelled by what Chinese media termed the "lobster craze" (lóngxiā rècháo, a wordplay on "Claw") [40][39]. During this phase, OpenClaw surpassed React (243,438 stars accumulated over 10 years) on March 3, 2026, at 250,829 stars, and was officially recognized as the most-starred software project on GitHub on the same day [2]. Growth averaged approximately 4,000 stars per day during this phase, reaching 335,000 by March 24 [1][40].

Phase 4: Stabilization (March 24 – present). Growth decelerated to under 1,000 stars per day, reaching 360,000+ stars on GitHub by late April 2026 [40][39][4]. This phase coincided with increasing media coverage of security vulnerabilities and the disclosure of the ClawHavoc supply chain campaign.

The growth trajectory is unusual: rather than following a simple post-viral decay curve, OpenClaw exhibited an abrupt late-January inflection, sustained acceleration through mid-February, and a later March reacceleration associated with Chinese adoption. This pattern suggests that the project's growth was not a single viral event but a compounding phenomenon in which each adoption wave created conditions for the next [19].

3.3 Governance Transitions

OpenClaw's governance has undergone three transitions in rapid succession:

Solo maintainer (November 2025 – January 2026). In its Clawdbot phase, Steinberger was the sole maintainer, making all architectural and release decisions.

Community-governed open source (January – February 2026). The rebrand to OpenClaw coincided with the opening of contributions. The contributor base grew from 1 to 1,800+ contributors by late April 2026 [19][4]. However, the project lacked formal governance structures—no steering committee, no security response team, and no code review requirements for skill submissions to ClawHub.

Foundation transition (February 2026 – present). Steinberger's hire by OpenAI in February 2026 [38] created uncertainty about the project's independence. In response, a foundation structure was announced to steward the project. Notably, this governance transition coincided with OpenClaw's peak security-disclosure burst: 9 entries were added in a 4-day window (March 18–21, 2026) [5]. Whether the transition affected security accountability remains unclear; the disclosure spike may reflect increased researcher scrutiny of a high-profile project as much as any governance gap.

Table 1: Timeline of OpenClaw milestones.

Date	Event	Description	Ref.
2025-11-24	Clawdbot launched	Peter Steinberger releases personal AI agent as weekend project; macOS-only, WhatsApp interface, Claude backend	[36]
2026-01-29	Rebranded to OpenClaw	Final rebrand after the brief Moltbot interim name; model-agnostic architecture, ClawHub skill marketplace, and multi-platform messaging support debut together	[1]
2026-01-30	Peak viral day	710 stars/hour (17,084/day) reported within the first 48 hours after the OpenClaw rebrand	[1][2]
2026-02-02	60,000 stars	60K reached within four days post-rebrand; Phase 1 (Viral Ignition) concludes	[1]
2026-02-07	ClawHavoc disclosed	Koi Security identifies 341 malicious skills among 2,857 on ClawHub (∼12%); subsequent reporting later places the total at 1,184	[7][8]
2026-02-12	190,000 stars	190K in 14 days; NanoClaw, ZeroClaw, PicoClaw, Nanobot forks emerge within a 14-day window	[1][19]
2026-02-15	Steinberger joins OpenAI	Creator hired by OpenAI; foundation governance announced for OpenClaw	[38]
2026-02-24	Surpassed Linux kernel	Crosses 224K+ stars and exceeds Linux's 218K stars (accumulated over 30 years)	[2][39]
2026-03-03	Surpassed React	Exceeds React's 243,438 stars (accumulated over 10 years)	[2]
2026-03-03	Most-starred repository	250,829 stars; officially the most-starred software project on GitHub	[2]
2026-03-18–21	Peak disclosure burst	9 new security-tracker entries logged in 4 days, including critical authentication bypasses	[5]
2026-03-24	China wave peak	335K stars; Chinese traffic +1,436% MoM; "lobster craze" phenomenon	[40][39]
Late Apr. 2026	Current	360K+ stars on GitHub; 3M+ MAU; 40K+ ClawHub skills; 1,800+ repo contributors	[4][40][1]

3.4 Naming and Identity

The project's naming history reflects its evolving identity. The original name "Clawdbot" was a portmanteau of "Claude" (the Anthropic LLM that served as its initial backend) and "bot." Following trademark concerns raised by Anthropic, the project was briefly renamed "Moltbot" on January 27, 2026, before settling on "OpenClaw" on January 29, 2026 [1]. The final rebrand signaled two shifts: the move to model-agnosticism (dropping the Claude-specific reference) and the adoption of "Open" to emphasize its open-source nature. The community has subsequently adopted the claw/lobster motif as a cultural identity marker, with derivative projects adopting names like NanoClaw, ZeroClaw, PicoClaw, and Dr. Claw [19].

3.5 Summary

OpenClaw's evolution from a weekend experiment to the most-starred software project in GitHub history occurred in under five months. This trajectory was enabled by three factors: (1) the accessibility of its messaging-platform interface, which lowered the barrier to autonomous AI from developers to general users; (2) the permissiveness of its architecture, which granted the LLM controller system-level OS access; and (3) the openness of its

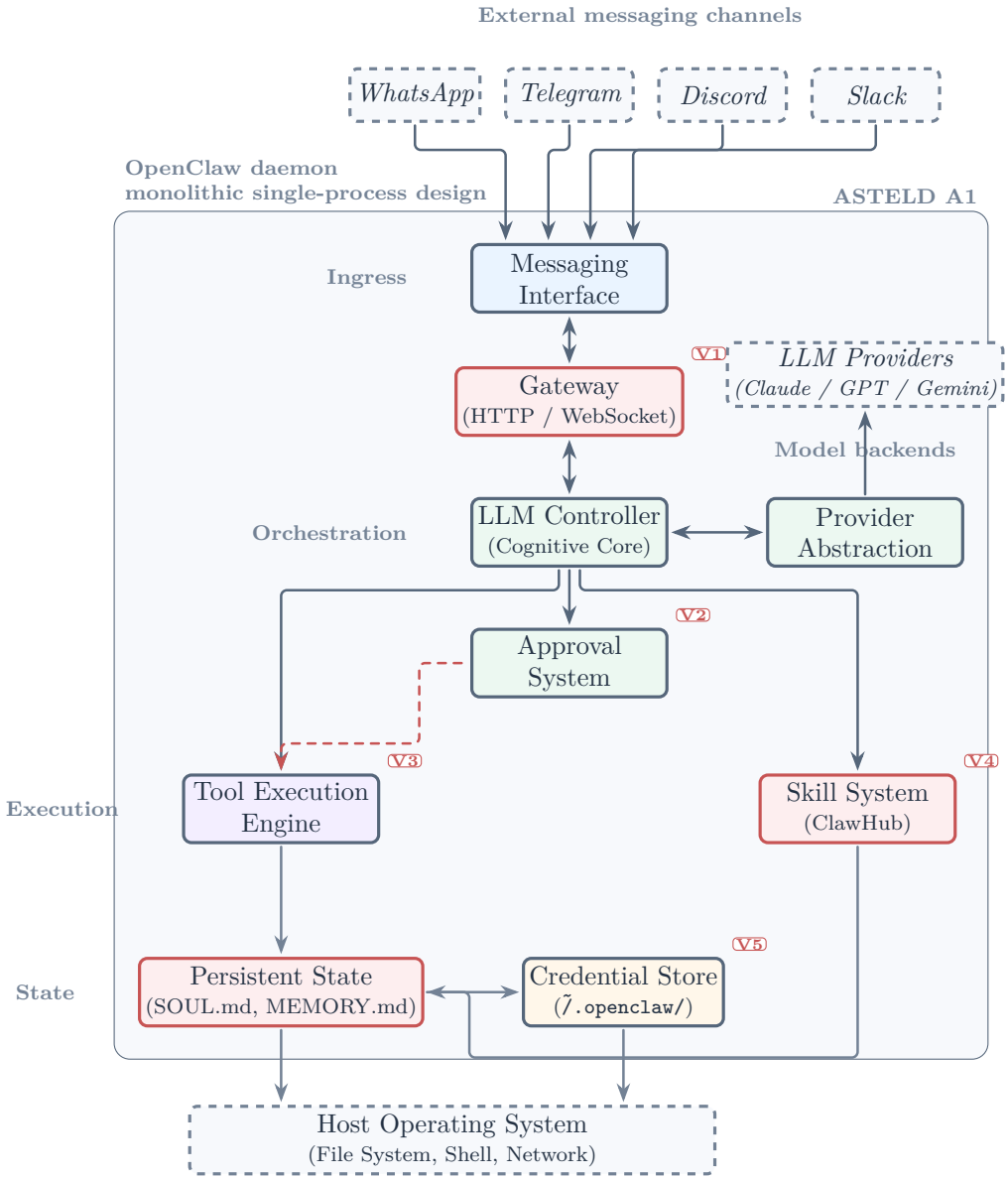


Figure 1: OpenClaw architecture. Red-tinted nodes mark components with documented security issues. V1 denotes gateway exposure via `0.0.0.0`; V2 approval-bypass CVEs; V3 privileged tool execution; V4 the ClawHub skill attack surface; and V5 plaintext or unencrypted credential storage [15][16][17].

extension ecosystem, which allowed anyone to publish skills with minimal vetting. As we show in the following sections, these same three factors created the conditions for OpenClaw's most serious security vulnerabilities.

4 System Architecture

This section provides a technical analysis of OpenClaw's architecture as of version 2026.3 (March 2026), drawing on official documentation [13], source code analysis, and third-party security assessments [15][16][17].

4.1 Architectural Overview

OpenClaw follows a **monolithic daemon architecture** (ASTELD category A1): a single long-running process that integrates the LLM controller, tool execution engine, skill loader, messaging interface, and local gateway into one TypeScript application. This design stands in contrast to the modular, microservice, or graph-based architectures adopted by competing frameworks (see Section 6).

The system comprises five principal components—Gateway, LLM Controller, Tool Execution Engine, Skill System, and Messaging Interface. Additional subcomponents (Provider Abstraction, Approval System, Persistent State, and Credential Store) are shown in Figure 1 and discussed in Section 5.

1. **Gateway**: An HTTP/WebSocket server that exposes the agent's capabilities to local and remote clients. By default, the gateway binds to `localhost` on a dynamically assigned port, but misconfigurations frequently result in binding to `0.0.0.0`, exposing the agent to the public internet [15][16][17].

2. **LLM Controller**: The cognitive core that receives user instructions (via messaging platforms or the gateway), maintains conversational context, formulates plans, and selects tools for execution. OpenClaw supports multiple LLM backends—including Anthropic Claude, OpenAI GPT-4, Google Gemini, and local models via Ollama—through a provider abstraction layer shown as a distinct subcomponent in Figure 1 [13].

3. **Tool Execution Engine**: Responsible for executing the actions selected by the LLM controller. Tools include shell command execution, file system operations, HTTP requests, and skill invocations. The engine operates with the full privileges of the user account under which the daemon runs—typically the user's primary login account [15].

4. **Skill System**: An extension mechanism that allows third-party developers to package tool definitions, prompt templates, and execution logic as installable "skills." Skills are distributed through ClawHub, an open marketplace that had grown to 40,000+ entries by April 2026 [40][1]. The skill system is the primary vector for the supply chain attacks described in Section 5.

5. **Messaging Interface**: Adapters that connect the daemon to messaging platforms (WhatsApp, Telegram, Discord, Slack). This component is architecturally significant because it means user instructions arrive through channels designed for human-to-human communication, not for machine control—a design choice with profound security implications [15].

4.2 Execution Model

OpenClaw employs a **single-agent sequential execution loop** (ASTELD category E1): observe → think → act → observe. The LLM controller receives an instruction, reasons about the appropriate action, invokes a tool, observes the result, and iterates until the goal is achieved or the user intervenes.

This execution model has three notable properties:

Unbounded iteration. Unlike frameworks that impose step limits or token budgets, OpenClaw's default configuration allows the agent to iterate indefinitely. Combined with

system-level tool access, this means a single malicious instruction can trigger an arbitrarily long chain of privileged actions [9].

Approval system. OpenClaw implements a command approval mechanism in which certain actions (e.g., shell commands, file deletions) are presented to the user for confirmation before execution. However, as documented in Section 5, this system suffers from TOCTOU (time-of-check-time-of-use) vulnerabilities: the command displayed to the user may differ from the command actually executed [10][5].

"Allow always" mode. Users can configure the approval system to automatically approve all actions of a given type. Microsoft's security assessment found that the majority of users default to this configuration for convenience, effectively operating at autonomy level L4 (delegated autonomy with no per-action oversight) despite the system's design for L2 (guided autonomy with approval) [15]. This behavioral pattern—users choosing convenience over security—is a recurring theme in the security literature on autonomous agents.

4.3 The Lethal Trifecta

As introduced in Section 2.3, Microsoft's security analysis [15][33] identifies a structural property of OpenClaw's architecture that it terms the **"lethal trifecta"**: the simultaneous presence of (1) access to private data on the user's device, (2) exposure to untrusted content from external sources (web pages, emails, documents), and (3) the ability to communicate with and act upon external services.

When all three conditions hold, prompt injection transitions from a content moderation problem to an **authorization bypass**. An attacker who can embed a malicious instruction in a web page, email, or document that the agent processes can cause the agent to:

- Read and exfiltrate private files (condition 1 enables access, condition 3 enables exfiltration)
- Install software or modify system configurations (condition 3 enables action)
- Persist the attack by modifying the agent's memory files (SOUL.md, MEMORY.md), ensuring the malicious behavior survives restarts [10][32]

This is not a bug in the traditional sense; it is an architectural property. The monolithic daemon design places the LLM controller, which processes both trusted instructions and untrusted content, in the same security domain as the tool execution engine, which has full system privileges. There is no privilege separation, no sandbox boundary, and no formal distinction between instruction and data at the architectural level [15][16].

4.4 State and Memory Management

OpenClaw maintains several forms of persistent state:

Conversational memory. The LLM controller maintains a context window of recent interactions. This is ephemeral and bounded by the LLM's context length.

Long-term memory. OpenClaw supports persistent memory through two mechanisms: (1) SOUL.md files that define the agent's personality and behavioral guidelines, and (2) MEMORY.md files that store accumulated knowledge. Both are stored as plaintext Markdown files in the user's home directory (`~/.openclaw/`) and are loaded into the LLM's context at startup [10].

Table 2: ASTELD profile of OpenClaw.

Axis	Category	Description
Architecture	A1	Monolithic daemon
Security	S2	Reactive, post-hoc patching
Tool Integration	T2	Plugin marketplace (ClawHub)
Execution	E1	Single-agent sequential loop
Level of autonomy	L2/L4	Designed for L2; users default to L4
Deployment	D1	Local-first personal agent

Credential storage. API keys, tokens, and authentication credentials are stored in plaintext in `~/.openclaw/openclaw.json` and environment files. No encryption, no OS keychain integration, and no access controls beyond filesystem permissions [17][35].

The security implications of this state management approach are significant. Because SOUL.md and MEMORY.md are loaded into the LLM context, an attacker who can modify these files—through a prompt injection, a malicious skill, or direct file access—can persistently alter the agent's behavior. This "memory poisoning" attack vector is documented in Section 5 and represents one of the most insidious threats to autonomous agent systems [10][32].

4.5 Deployment Topology

OpenClaw is designed as a **local-first personal agent** (ASTELD category D1): the daemon runs on the user's device, and the primary interaction occurs through messaging platforms. However, the reality of deployment diverges significantly from the design intent:

- **Approximately 500,000+ instances** were reported as internet-facing in late-March 2026 live checks, with **15,000+** directly exploitable via known remote-code-execution paths [41][42].
- **More than 30,000 exposed instances** were observed with material security risks during the same reporting window [42].
- The gateway, intended for localhost communication, is frequently exposed to the public internet due to Docker misconfiguration, cloud deployment, or explicit user choice [15].

This gap between designed topology (local-first, personal) and actual topology (internet-exposed, multi-user) is a recurring source of vulnerabilities. The security model assumes a trusted local environment; the deployment reality is an untrusted networked environment.

4.6 Architectural Comparison via ASTELD

To contextualize OpenClaw's architectural decisions, we apply the ASTELD classification framework introduced in this paper (see Section 6 for full framework description and cross-platform comparison). OpenClaw's ASTELD coordinates are:

This profile is unique among the eight frameworks analyzed in Section 6. No other framework combines A1 (monolithic) with T2 (marketplace) and D1 (local-first). The combination explains both OpenClaw's distinctive strengths—simplicity, accessibility, low setup friction—and its distinctive vulnerabilities—no privilege separation, marketplace as attack surface, local daemon with internet exposure.

4.7 Summary

OpenClaw's architecture is a study in trade-offs. The monolithic daemon design enables the simplicity and accessibility that drove viral adoption, but it also creates a security surface that is fundamentally different from—and more dangerous than—that of modular or graph-based frameworks. The lethal trifecta is not a flaw to be patched; it is a consequence of architectural decisions that prioritized user experience over security isolation. As we demonstrate in the next section, the vulnerabilities that follow are largely predictable from this architectural profile—rooted in design decisions rather than implementation accidents.

5 Security and Privacy Analysis

OpenClaw's security trajectory is unlike that of any prior open-source project. Within five months of its viral breakout, public trackers documented 100+ tracked advisories and 10+ published CVEs, while broader vendor summaries reported larger tallies of tracked vulnerability records under different aggregation rules; these sources also converged on the presence of several critical-severity vulnerabilities (CVSS $\geq$ 9.0), dozens of additional high-severity findings, and a 4-day burst in which 9 new security-tracker entries were logged between March 18 and March 21, 2026 [9][5][6]. At the time of writing, approximately 500,000+ OpenClaw instances had been reported as internet-facing, more than 30,000 were observed with material security risks, and 15,000+ were directly exploitable via known RCE paths [41][42].

This section presents a six-category vulnerability taxonomy synthesized from CVE databases, five institutional security assessments [15][16][17][18][8], and academic research [9][10][43]. We then identify four canonical attack chains that compose vulnerabilities across categories, and conclude with an assessment of the defense landscape.

5.1 Vulnerability Taxonomy

We classify OpenClaw vulnerabilities into six categories based on attack vector, impact, and architectural root cause. The categories are not mutually exclusive: real-world attacks frequently chain vulnerabilities across categories (see Section 5.2).

5.1.1 Category A: Authentication and Session Trust Abuse

The architectural root cause is OpenClaw's trust assumption that all localhost connections are legitimate—an assumption that does not hold when the gateway is internet-exposed or when browser-based attacks target localhost.

Three critical CVEs define this category. CVE-2026-25253 (ClawJacked/ClawBleed, CVSS 8.8) enables a malicious webpage to initiate cross-site WebSocket hijacking against the localhost gateway; because the browser's Same-Origin Policy does not block WebSocket connections to localhost, and OpenClaw's rate limiter exempts 127.0.0.1, an attacker can brute-force the gateway port and exfiltrate authentication tokens, achieving full agent takeover [34][44]. CVE-2026-22172 (CVSS 9.9) allows a WebSocket client to self-declare administrative scopes, bypassing authentication entirely—the gateway trusts the client's scope assertion without verification [5]. CVE-2026-32922 (Critical) enables a single API

Table 3: OpenClaw security vulnerability taxonomy.

Cat.	Category Name	Root Cause	Representative Evidence	CVSS / Severity	Count
A	Authentication & Session Trust Abuse	Trust-boundary failures; improper scope/token binding in WebSocket-based control paths	CVE-2026-25253 (ClawJacked) CVE-2026-22172 (Scope Self-Decl.) CVE-2026-32922 (Pairing Escalation)	8.8–9.9 High–Crit.	3+
B	Approval System Bypass	Display-time vs. execution-time inconsistency (TOCTOU analog)	CVE-2026-29607 (Allow-Always) CVE-2026-28460 (Line Continuation) CVE-2026-34426 (Env Var Injection) CVE-2026-32065 (Token Mismatch)	6.5–8.8 Med.–High	4
C	OS-Specific Escapes	Shell escaping failures across platforms	CVE-2026-22179 (macOS cmd subst.) CVE-2026-22176 (Windows .cmd)	High 7.8	2
D	Prompt Injection	No architectural separation of instructions from data in LLM context	Direct, indirect (IDPI/XPIA), memory poisoning, guidance injection, cross-agent propagation	—	5 subtypes
E	Supply Chain (ClawHavoc)	Unvetted marketplace; weak publisher vetting; unvetted publishing pipeline	1,184 malicious skills identified Trojan/OpenClaw.PolySkill	—	1,184 skills
F	Infrastructure Misconfig.	Local-dev defaults deployed at internet scale	CVE-2026-32018 (No file locking) 500K+ internet-facing instances 15K+ known-RCE exploitable Plaintext credential storage	6.6 — — —	3+

Representative evidence items include CVEs where assigned and non-CVE records where the category is documented through tracker entries, marketplace audits, or infrastructure exposure reports.

call to escalate a pairing token into full administrative control with remote code execution capabilities [35].

The common thread is a **trust boundary violation**: mechanisms designed for trusted local environments are deployed in untrusted networked environments.

5.1.2 Category B: Approval System Bypass

OpenClaw's command approval system is its primary human-in-the-loop security mechanism—and it has been bypassed through four distinct CVEs, each exploiting a different inconsistency between the approval display path and the execution path.

CVE-2026-29607 exploits the fact that "allow always" persists at the wrapper command level, not the inner command; an attacker can swap the inner payload after the wrapper has been approved, achieving persistent RCE without re-prompting [5]. CVE-2026-28460 demonstrates that shell line-continuation characters bypass the command allowlist entirely [5]. CVE-2026-34426 shows that inconsistent environment variable normalization between the approval and execution paths enables environment variable injection without triggering the approval prompt [5]. CVE-2026-32065 (CVSS 5.7) reveals a mismatch between how command tokens are displayed during approval and how they are resolved

at execution time [10].

The architectural insight is that the approval system suffers from a class of vulnerability analogous to the TOCTOU (time-of-check-time-of-use) problem in operating system security [10]. The "check" (displaying the command for approval) and the "use" (executing the command) operate on different representations of the same action, creating a semantic gap that attackers can exploit. No formal specification of approval correctness exists for any agent system, making this an open research problem (see RQ3, Section 8).

5.1.3 Category C: Operating System-Specific Escapes

Platform-specific shell escaping failures create additional attack vectors. CVE-2026-22179 (High) allows command substitution syntax inside double-quoted strings to bypass the command allowlist on macOS [5]. CVE-2026-22176 (CVSS 7.8) exploits unescaped environment variables in scheduled task `.cmd` scripts on Windows, where characters such as `&`, `|`, and `^` are interpreted as command separators [5].

These vulnerabilities, while platform-specific, illustrate a broader point: OpenClaw's tool execution engine must correctly handle shell semantics across every supported operating system—a combinatorially complex problem that is intrinsically difficult to solve in a monolithic architecture where the LLM controller generates shell commands directly.

5.1.4 Category D: Prompt Injection

Prompt injection in autonomous agents is qualitatively different from prompt injection in chatbots. In a chatbot, a successful injection may cause the model to generate inappropriate text. In an agent with system-level tool access, a successful injection can cause the execution of arbitrary shell commands, the exfiltration of private data, and the persistent modification of the agent's behavior.

We identify five sub-types of prompt injection relevant to OpenClaw:

Direct injection: The user types malicious instructions that override the system prompt, causing the agent to execute unintended operations [9].

Indirect injection (IDPI/XPIA): Malicious instructions are embedded in content—web pages, emails, documents, MCP tool outputs—that the agent processes during legitimate task execution. Because the LLM processes instructions and data in the same context window with no architectural separation, the embedded instructions can hijack the agent's reasoning [10][32][45].

Memory poisoning: An attacker modifies the agent's persistent memory files (SOUL.md, MEMORY.md) to inject instructions that alter the agent's behavior. Because these files are loaded into the LLM context at every startup, the injected behavior persists across restarts and may include time-delayed activation triggers [10][32]. This attack vector is unique to agent systems with persistent state and represents one of the most insidious threats identified in the literature.

Guidance injection: Attacker-controlled instructions in ClawHub skill descriptions or MCP server tool outputs alter the agent's reasoning during skill execution. Unlike indirect injection through user data, guidance injection operates through the tool layer itself—the agent trusts tool outputs as authoritative, creating a privilege escalation path [46][8].

Cross-agent propagation: In multi-instance deployments or social-agent networks (such as Moltbook), a malicious prompt in shared content can reach multiple agents

simultaneously, causing coordinated compromise [13][12]. This vector remains understudied; only one empirical analysis exists [13], and no formal threat model for agent-to-agent infection has been proposed (see RQ6, Section 8).

The key insight, shared by Microsoft [15], CrowdStrike [17], and academic researchers [9][10], is that prompt injection in agentic systems is an **authorization problem**, not a content moderation issue. Tool access is the amplifier: systems without external tool access show minimal successful injection outcomes. The architectural separation of instruction processing from data processing remains unsolved (see RQ1, Section 8).

5.1.5 Category E: Supply Chain Attacks (ClawHavoc)

The ClawHavoc campaign, disclosed in February 2026, represents the first large-scale supply chain attack targeting an AI agent skill marketplace [7][46][8]. The campaign's root cause was ClawHub's minimal vetting requirement: publishing a skill required only a GitHub account that was one week old.

The attack was executed through multiple sub-campaigns. The primary ClawHavoc campaign used fake error messages to trick users into pasting base64-encoded commands that installed Atomic Stealer (AMOS)—malware that harvests browser credentials, cryptocurrency wallets, and system data [7]. The AuthTool sub-campaign deployed dormant payloads activated by specific user prompts, establishing persistent reverse shells [46]. The Hidden Backdoor campaign disguised itself as an Apple Software Update during skill installation, creating an encrypted tunnel to attacker infrastructure [8]. A benign-looking weather assistant skill was found to steal API keys from the `.clawdbot/.env` file [8].

The scale of the campaign is striking. Koi Security's initial audit identified 341 malicious skills among 2,857 total (approximately 12%) [7]. Bitdefender's expanded scan found 824+ among 10,700+ skills [46]. Subsequent reporting later placed the total at 1,184 malicious skills, and Koi Security assigned the classification Trojan/OpenClaw.PolySkill [7][8]. A single threat actor ("hightower6eu") was responsible for 354 of these skills [7].

ClawHavoc established a new threat class: **AI agent supply chain attacks through natural language payloads**. Unlike traditional software supply chain attacks (e.g., npm, PyPI package poisoning) where malicious code can be detected through static analysis, agent skill poisoning can operate entirely through natural language instructions that the LLM interprets as commands. This makes conventional code scanning tools, including VirusTotal (which OpenClaw adopted post-ClawHavoc), fundamentally insufficient for detection [8][33].

5.1.6 Category F: Infrastructure Misconfigurations

Default configurations designed for local development are routinely deployed at internet scale. The gateway binds to `0.0.0.0` (exposing all network interfaces) in many deployment configurations; by late March 2026, public reporting described approximately 500,000+ internet-facing instances, more than 30,000 with material security risks, and 15,000+ directly exploitable through known RCE paths [41][42]. API keys and credentials are stored in plaintext in `~/.openclaw/openclaw.json` and `.env` files with no encryption or OS keychain integration [17][35]. Authentication tokens transmitted in URL query parameters are leaked to browser history and server logs [5]. The absence of file locking (CVE-2026-32018, CVSS 6.6) causes concurrent registry operations to produce orphaned containers and inconsistent state [5].

5.2 Attack Chain Taxonomy

Individual vulnerabilities rarely exist in isolation. We identify four canonical attack chains that combine vulnerabilities across the categories defined above.

Chain 1: Browser-Initiated Full Takeover. A malicious webpage (Category A: ClawJacked) initiates a WebSocket brute-force against the victim's localhost gateway → exfiltrates the authentication token → achieves full agent control → executes arbitrary shell commands. This chain requires only that the victim visits a webpage while OpenClaw is running—no social engineering, no malware installation, no user interaction beyond the initial page visit [34].

Chain 2: Supply Chain Persistent Compromise. A malicious ClawHub skill (Category E) installs and tampers with persistent instruction files such as SOUL.md or MEMORY.md (Category D: memory poisoning), creating a delayed behavioral backdoor. When the trigger condition is later met, the poisoned memory can steer the agent toward credential exfiltration or attacker-directed command execution; the compromise persists across restarts because the poisoned memory is reloaded at startup [46][8].

Chain 3: Indirect Injection Escalation. A poisoned web page, email, or document (Category D: indirect injection) is ingested by the agent during legitimate task execution → the embedded instructions override the agent's reasoning → the agent exploits an approval bypass (Category B) or executes a shell command directly → data is exfiltrated to an attacker-controlled server [10][32].

Chain 4: Cross-Agent Propagation. A poisoned message introduced into a shared feed or Moltbook interaction (Category D: cross-agent exposure) can be processed by multiple agent instances. If those agents are configured to write untrusted content into persistent memory, the same malicious instruction may propagate into separate memory states, creating the potential for repeated unauthorized actions across multiple agents [13].

5.3 Institutional Security Assessments

The convergence of institutional assessments regarding OpenClaw is noteworthy. Five major security organizations independently evaluated the platform and reached remarkably consistent conclusions, suggesting that its vulnerabilities are systemic rather than superficial.

Microsoft Security emphasized the risks to host environments, recommending that users avoid deploying OpenClaw with primary work or personal accounts and proposing the use of dedicated virtual machines with non-privileged credentials as a minimum precaution [15]. CertiK identified a fundamental architectural contradiction, characterizing the vulnerability profile as a flaw designed for trusted local use, but deployed at internet scale, and advocated for mobile-OS-style permission declarations for all integrated skills [16].

From a threat-vector perspective, CrowdStrike classified OpenClaw as an AI super agent requiring intensive security scrutiny, warning that its extensive tool access serves as a primary amplifier for prompt injection attacks [17]. Trend Micro focused on enterprise governance, advising Chief Information Security Officers (CISOs) to apply the same level of security rigor to OpenClaw deployments as is standard for production-grade servers [18]. Complementing these views, Koi Security provided critical insights into supply chain integrity, documenting the ClawHavoc activity and the proliferation of malicious skills designed to exploit the platform's execution model [8].

The institutional consensus is definitive: OpenClaw's security challenges are structural, not incidental. These vulnerabilities are direct consequences of foundational architectural

decisions—specifically the monolithic daemon design, the unvetted marketplace, plaintext credential storage, and the localhost trust assumption—rather than implementation bugs that can be resolved through isolated patches.

5.4 Defense Landscape

The defense landscape spans three tiers of intervention:

Tier 1: OpenClaw's own mitigations. Post-disclosure releases addressed several concrete flaws, including the one-click RCE path fixed in version 2026.1.29, later hardening of gateway authentication and localhost handling, and subsequent closures for multiple March-reported attack paths [47][5]. After ClawHavoc, the ecosystem also moved toward malware screening and user reporting for skill submissions [5]. These mitigations are primarily reactive: they reduce exposure to known exploit paths, but do not by themselves eliminate the underlying architectural weaknesses.

Tier 2: Community and derivative approaches. NanoClaw adopts per-agent containerized isolation, limiting each agent's access to explicitly mounted resources [19]. Other derivative or adjacent systems propose stronger guardrails such as owner-governed policy controls, sandboxed execution, and architectural separation between model interpretation and tool invocation [45][19].

Tier 3: Third-party security tools. Third-party defenses aim to monitor and constrain agent behavior at runtime. For example, Prompt Security's AI Gateway inspects traffic between AI applications and MCP servers and enforces allow/block policies on risky interactions [45]. Enterprise hardening guidance further recommends container or VM isolation, secrets minimization, memory-file integrity monitoring, and outbound connection monitoring for exfiltration detection [15][17].

However, these defenses mostly reduce exploitability or blast radius rather than removing the deeper failure modes identified in our taxonomy, including localhost trust-boundary violations, approval-time vs. execution-time semantic mismatch, and the lack of reliable separation between instructions and untrusted data in agent context.

5.5 Summary

OpenClaw's security landscape reveals a fundamental mismatch between deployment scale and security maturity. The six vulnerability categories—authentication hijacking, approval bypass, OS-specific escapes, prompt injection, supply chain attacks, and infrastructure misconfigurations—are not independent; they compose into multi-stage attack chains that amplify individual vulnerabilities into full system compromise. The institutional consensus, the volume of CVEs, and the architectural analysis all point to the same conclusion: the challenges are systemic, rooted in architectural decisions that cannot be fully remediated through patching alone. As we show in the next section, these challenges are not unique to OpenClaw but are shared, to varying degrees, across the autonomous agent ecosystem.

6 Ecosystem Comparison and the ASTELD Classification Framework

This section positions OpenClaw within the broader landscape of autonomous AI agent frameworks through a systematic comparison of eight platforms, organized around the ASTELD classification framework introduced in this paper.

6.1 Motivation and Scope

The autonomous AI agent ecosystem has expanded rapidly since AutoGPT's viral release in March 2023 [22]. By April 2026, eight major open-source frameworks compete across different architectural philosophies, security models, and target use cases: OpenClaw [13], Hermes [48], AutoGPT [22], CrewAI [24], LangGraph [21], n8n [49], Dify [50], and AutoGen/Microsoft Agent Framework [23][51]. These frameworks differ significantly in architecture, tool integration, level-of-autonomy mechanisms, and deployment strategies, making direct comparison challenging. Hermes is especially relevant because it targets the same broad personal-agent space as OpenClaw but shifts the deployment model toward a server-resident D2 profile and emphasizes a built-in learning loop with persistent cross-session memory [48]. While prior work has explored aspects of these systems, there remains a need for a more standardized, multi-dimensional comparison. Prior analyses have provided valuable insights through framework-specific studies [9][10][11], pairwise comparisons [52][53], and practical rankings or benchmarks [54], but they typically focus on specific perspectives or evaluation criteria. To address this need, we introduce the ASTELD classification framework, which enables systematic comparison across multiple architectural and functional dimensions.

6.2 The ASTELD Classification Framework

We propose ASTELD, a six-axis taxonomy for classifying autonomous AI agent platforms, where each axis is defined by observable system properties and assigned using explicit criteria. The name is an acronym for its axes: **A**rchitecture pattern, **S**ecurity posture, **T**ool integration model, **E**xecution paradigm, **L**evel of autonomy and human control, and **D**eployment topology. The framework was designed to satisfy four properties: (1) **discriminative**—every axis separates at least two frameworks, and frameworks can be consistently mapped to distinct dominant profiles even when some axes require range-valued entries under common configurations; (2) **comprehensive**—the axes jointly capture key technical, security, and operational properties; (3) **empirically grounded**—every axis value is derived from observed framework properties, not theoretical design space; and (4) **extensible**—derivatives and future frameworks can be positioned without restructuring the taxonomy.

ASTELD builds upon and extends six prior taxonomies identified in the literature, each of which addresses a subset of the classification problem: Sapkota et al.'s conceptual distinction between AI agents and agentic AI [30], the six-component agent anatomy of V et al. [27], Abou Ali et al.'s dual-paradigm framework [55], Interface EU's five-level autonomy classification [31], Piccialli et al.'s industry-focused autonomy progression [56], and the Frontiers data leakage taxonomy [57].

The six axes are defined as categorical dimensions. Each framework is assigned a dominant value per axis where possible; when common real-world configurations materially

span multiple categories, we retain an explicit range or transition notation rather than forcing an artificial single-value assignment.

Axis 1: Architecture Pattern (A). Classifies the structural organization of the agent system into five categories: A1 (Monolithic Daemon)—a single long-running process handling all agent functions; A2 (Visual Workflow)—node-graph editor as primary interface with execution following a visual DAG; A3 (Composable Library)—agent primitives provided as a library for developer composition; A4 (Graph-Based Runtime)—stateful directed graph with cycles, checkpointing, and replay; A5 (Event-Driven Multi-Agent)—event bus architecture with asynchronous message passing across multiple agents.

Axis 2: Security Posture (S). Classifies the maturity of built-in security mechanisms, adapted from software security maturity models (BSIMM, OWASP SAMM), into four levels: S1 (Minimal)—no built-in security boundary, relying on OS-level permissions; S2 (Reactive)—post-hoc patches, approval prompts, allowlists; S3 (Framework-Level)—middleware permissions, tool-level access control, code execution sandboxing; S4 (Enterprise-Grade)—RBAC, SSO/SAML, audit logging, SIEM integration, air-gapped deployment support.

Axis 3: Tool Integration Model (T). Classifies how agents discover and invoke external tools: T1 (Static Configuration)—tools predefined at setup; T2 (Plugin Marketplace)—install-time selection from a curated or open marketplace; T3 (Protocol-Based MCP)—standardized Model Context Protocol for runtime tool discovery and invocation; T4 (Connector Ecosystem)—large library of pre-built connectors in an iPaaS-style model.

Axis 4: Execution Paradigm (E). Classifies computation structure: E1 (Single-Agent Loop)—observe-think-act cycle with sequential execution; E2 (Multi-Agent Conversation)—structured message exchange among multiple agents; E3 (Stateful Graph Execution)—state machine with branching, loops, and checkpoints; E4 (Event-Driven Pipeline)—trigger-node chain with workflow-level parallelism.

Axis 5: Level of autonomy and human control (L). Classifies the level of autonomous decision authority and human involvement, adapted from Interface EU's five-level model [31]: L1 (Tool Mode)—every action requires explicit user instruction; L2 (Guided Autonomy)—the agent proposes actions, and a human approves each one; L3 (Bounded Autonomy)—the agent executes within a predefined scope; L4 (Delegated Autonomy)—the agent decomposes and executes complex goals end-to-end; L5 (Full Autonomy)—continuous operation without human oversight. To avoid conflict with the first Architecture Pattern (A), the sublevels of level of autonomy and human control use the prefix L (Level).

Axis 6: Deployment Topology (D). Classifies where the agent runtime executes: D1 (Local-First Personal)—runs on user's device with local file/system access; D2 (Self-Hosted Server)—runs on user-managed server; D3 (Cloud-Managed)—vendor-hosted agent-as-a-service; D4 (Library-Embedded)—agent logic embedded in user's application code.

6.3 Framework Mapping

Table 4 presents the complete ASTELD mapping for all eight frameworks.

We acknowledge that some frameworks span multiple categories depending on configuration and usage context; assignments reflect their primary design and typical deployment.

OpenClaw's level of autonomy carries a notable caveat: the system is designed for L2

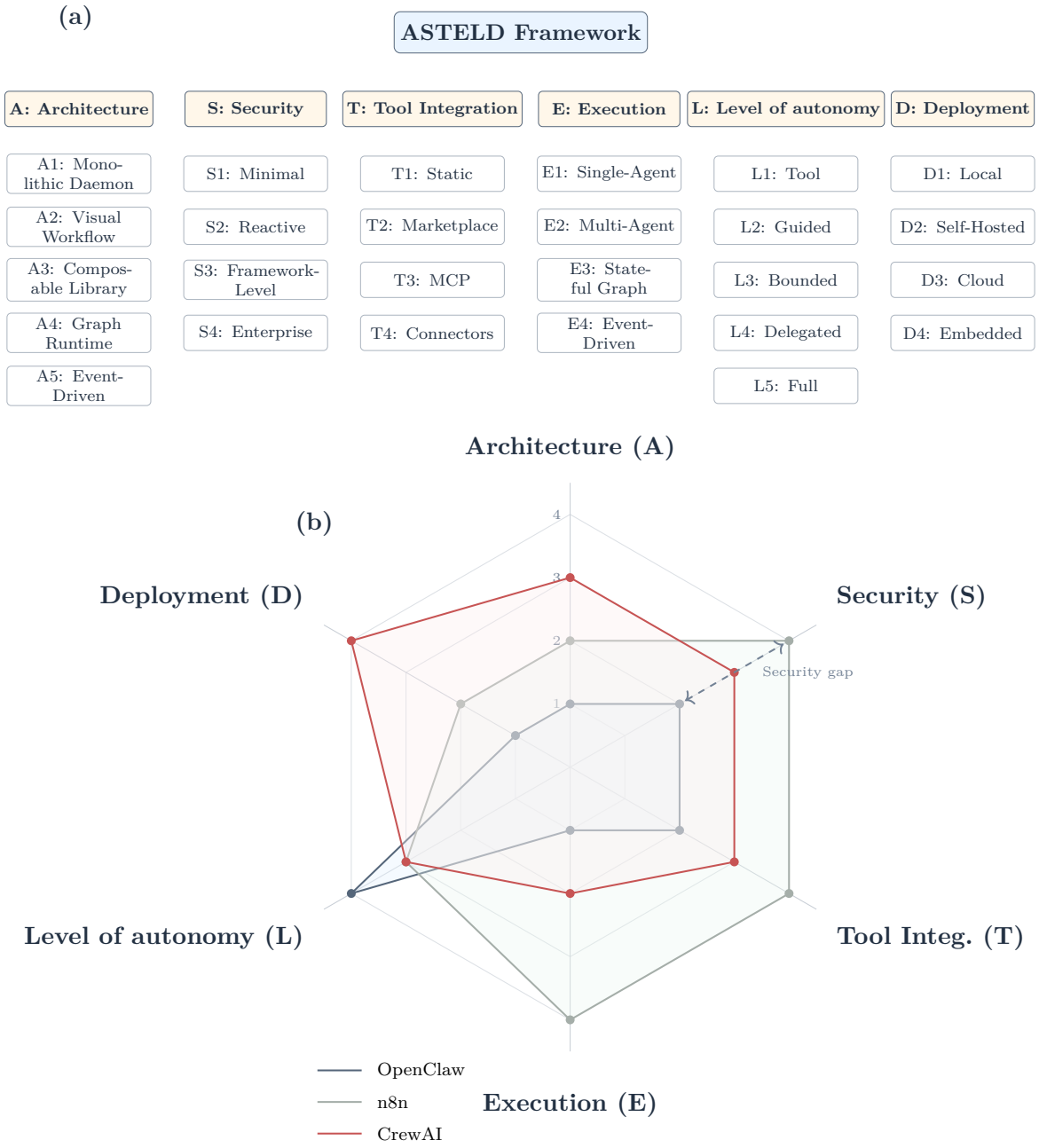


Figure 2: ASTELD taxonomy and representative profiles. Panel (a) lists the axes and categories. Panel (b) plots OpenClaw, n8n, and CrewAI.

(guided autonomy with per-action approval), but the majority of users configure "allow always" mode for convenience, effectively operating at L4 (delegated autonomy) with an S2 (reactive) security posture [15]. Hermes shows a related but distinct pattern: it remains architecturally monolithic (A1/E1) yet pushes toward a more server-resident and security-hardened S3/D2 profile through containerized terminal backends, persistent memory, and protocol-based extensibility [48]. These classifications reflect typical usage patterns rather than strictly enforced system constraints.

Table 4: ASTELD classification of eight autonomous AI agent platforms.

Framework	A	S	T	E	L	D	Stars	Primary Use Case
OpenClaw	A1	S2	T2	E1	L2/ L4†	D1	360K+	Personal AI assistant
Hermes	A1	S3	T3	E1	L3/ L4	D2	117K+	Server-resident personal agent
AutoGPT	A2	S2	T2	E1→ E4	L4	D2	180K+	Autonomous research
CrewAI	A3	S3	T3	E2	L3	D4	50K+	Multi-agent collaboration
LangGraph	A4	S3	T3	E3	L2– L3	D3/ D4	30K+	Stateful workflows
n8n	A2	S4	T4	E4	L3	D2/ D3	180K+	Workflow automation
Dify	A2	S4	T3	E4	L3	D2/ D3	130K+	Production AI apps
AutoGen / MS AF	A5	S3	T4	E2	L3– L4	D4	50K+	Research collaboration

† OpenClaw is designed for L2 (per-action approval), but many users configure "allow always," effectively operating at L4 (delegated autonomy) [15]. Axis definitions: A = Architecture Pattern, S = Security Posture, T = Tool Integration Model, E = Execution Paradigm, L = Level of autonomy, D = Deployment Topology. Star counts were refreshed from GitHub repository pages in late April 2026 [4][48][22][58][59][60][61][62].

6.4 Cross-Cutting Patterns

The ASTELD mapping suggests three structural patterns that extend beyond individual frameworks, based on the configurations summarized in Table 4.

Pattern 1: The Security-Accessibility Diagonal. Plotting Security (S) against Deployment (D) across the surveyed frameworks, a general inverse trend can be observed. D1 (local-first) platforms tend to align with S1–S2 (minimal to reactive security), with OpenClaw as a representative example. D2/D3 (server/cloud) platforms are more often associated with S3–S4 (framework to enterprise-level security), as seen in n8n and Dify. D4 (library-embedded) platforms typically correspond to S3 (framework-level security), where responsibility is partially delegated to developers.

Notably, none of the surveyed frameworks occupy the combination of D1 + S4 (local-first deployment with enterprise-grade security). While this observation is limited to the current sample, it highlights a potentially underexplored design space. In practice, users seeking locally running personal assistants may encounter more limited security features, whereas systems offering stronger security guarantees are more commonly deployed in server- or cloud-based settings. This trade-off appears in the current ecosystem, though further evidence would be needed to determine whether it reflects a fundamental constraint or a transient design choice.

Pattern 2: Execution-Architecture Coupling. Architecture pattern appears closely associated with execution paradigm. For example, A1 (monolithic) systems are typically paired with E1 (single-agent loops), A2 (visual workflow) with E4 (event-driven pipelines), A3 (composable libraries) with E2 (multi-agent interaction), A4 (graph runtimes) with E3 (stateful graph execution), and A5 (event-driven multi-agent systems) also with E2. This alignment suggests that architectural design choices often influence, or co-evolve with, execution models. However, this relationship is empirical rather than strictly deterministic, and alternative pairings may be possible.

Pattern 3: Convergence with Persistent Differentiation. Several frameworks

exhibit convergence toward a common set of capabilities, including model-agnostic design, support for tool integration protocols (e.g., MCP), visual workflow construction, and human-in-the-loop control mechanisms. At the same time, meaningful architectural differences remain. The ecosystem can be broadly interpreted as forming three functional groupings: personal assistant-oriented systems (e.g., OpenClaw, Hermes), multi-agent orchestration frameworks (e.g., CrewAI, LangGraph, AutoGen), and workflow automation platforms (e.g., n8n, Dify). While these groupings are not mutually exclusive, they reflect typical usage patterns observed in the current landscape. Cross-group competition appears more limited than within-group variation, although this characterization is based on a small set of representative systems [52][48].

6.5 Comparative Analysis by Dimension

Architecture. OpenClaw's monolithic daemon (A1) is distinctive in using messaging platforms as its primary user interface, which may lower the barrier to entry for non-technical users but can introduce additional challenges for enforcing security boundaries. Hermes occupies a nearby architectural niche but relocates the agent to a server-resident D2 deployment with persistent memory, MCP extensibility, and a built-in learning loop, making it a more direct competitor in the personal-agent lane than the workflow-centric or library-centric alternatives [48]. AutoGPT, n8n, and Dify share the visual workflow pattern (A2), although their underlying system designs differ; notably, n8n's node-based canvas predates AI agent frameworks and originates from general workflow automation [49]. LangGraph's graph-based runtime (A4) enables expressive control flow, including cycles and stateful execution, but may require greater familiarity with graph-based abstractions [21].

Security. The surveyed frameworks exhibit a range of security capabilities. n8n and Dify provide features such as RBAC, SSO/SAML, audit logging, and support for controlled deployment environments, reflecting their positioning toward enterprise use cases [49][50]. In contrast, OpenClaw is characterized here as S2 (reactive), relying more heavily on post-hoc controls and user configuration. While OpenClaw has a large deployment footprint, this comparison highlights a potential gap between deployment scale and the maturity of built-in security mechanisms. This observation is based on available feature sets and reported configurations rather than a formal security evaluation.

Tool Integration. The Model Context Protocol (MCP) is increasingly adopted across frameworks, with several platforms supporting protocol-based tool integration [21][49][50]. OpenClaw follows a T2 (marketplace-based) approach, enabling rapid expansion of available tools. Prior work has noted that marketplace-based ecosystems may introduce risks if contributions are not systematically vetted [7][8]. The observed shift toward protocol-based integration (T3) in some systems may reflect an effort to standardize and better control tool invocation, although both models continue to coexist in practice.

Human-in-the-Loop. Human-in-the-loop (HITL) mechanisms vary in granularity and integration. LangGraph provides support for checkpointing, state inspection, and intervention at arbitrary points in execution graphs [21]. n8n supports human intervention through workflow nodes, enabling approval steps and user input routing within pipelines [49]. OpenClaw includes an approval-based mechanism intended to gate actions, although its effectiveness depends on user configuration and system context (see Section 5). Overall, HITL support is present across frameworks but differs in depth and flexibility.

License and Governance. The surveyed frameworks adopt a range of licensing

Table 5: Agent-platform selection matrix for practitioners.

Use Case	Recommended	Rationale
Personal AI via messaging	OpenClaw	Only native WhatsApp/Telegram/Discord integration
Multi-agent collaboration	CrewAI	Purpose-built for role-based orchestration; 5.76x faster than LangGraph [52]
Complex stateful workflows	LangGraph	Most expressive graph control flow; best HITL; checkpoint/resume [21]
Visual no-code agents	n8n or Dify	400–600+ integrations; enterprise RBAC/SSO [49][50]
Autonomous research	AutoGPT	Pioneer in autonomous task loops; visual workflow builder [22]
Enterprise multi-agent	MS Agent Framework	Azure integration; 1,000+ connectors via Semantic Kernel [51]
Maximum security	n8n (self-hosted) or TrustClaw	RBAC/SSO/audit/SIEM; or OpenClaw fork with OAuth + sandbox [49][19]

models. OpenClaw, CrewAI, LangGraph, and AutoGen use permissive MIT licenses, while Dify adopts Apache 2.0. AutoGPT applies a Polyform Shield License to parts of its platform, and n8n uses a fair-code model (Sustainable Use License) alongside commercial offerings. These choices reflect differing approaches to ecosystem growth, monetization, and long-term maintenance. While permissive licenses may facilitate adoption, they do not inherently provide mechanisms for coordinated governance or resource allocation, which may influence sustainability considerations.

6.6 Framework Selection Matrix

Based on the comparative analysis, we propose a selection matrix for practitioners:

6.7 Summary

The ASTELD framework offers a structured lens for understanding the autonomous AI agent landscape. A primary contribution is making explicit the trade-offs that are otherwise implicit in each framework's design: OpenClaw's accessibility comes at the cost of security; n8n's enterprise security comes at the cost of personal-assistant simplicity; LangGraph's expressiveness comes at the cost of accessibility. No surveyed framework simultaneously achieves strong performance across all six axes, and the empty quadrant (D1 + S4) highlights an unsolved design challenge. This analysis is based on a limited set of representative frameworks and reflects dominant configurations rather than all possible system variants. The derivative ecosystem, examined in the next section, represents the market's attempt to fill these gaps through specialization rather than convergence.

7 Adoption, Social Impact, and the Derivative Ecosystem

OpenClaw's adoption trajectory is unprecedented in open-source history, both in its growth and in the breadth of its social impact. This section examines three interrelated phenomena: the quantitative growth dynamics, the sociotechnical implications of mass

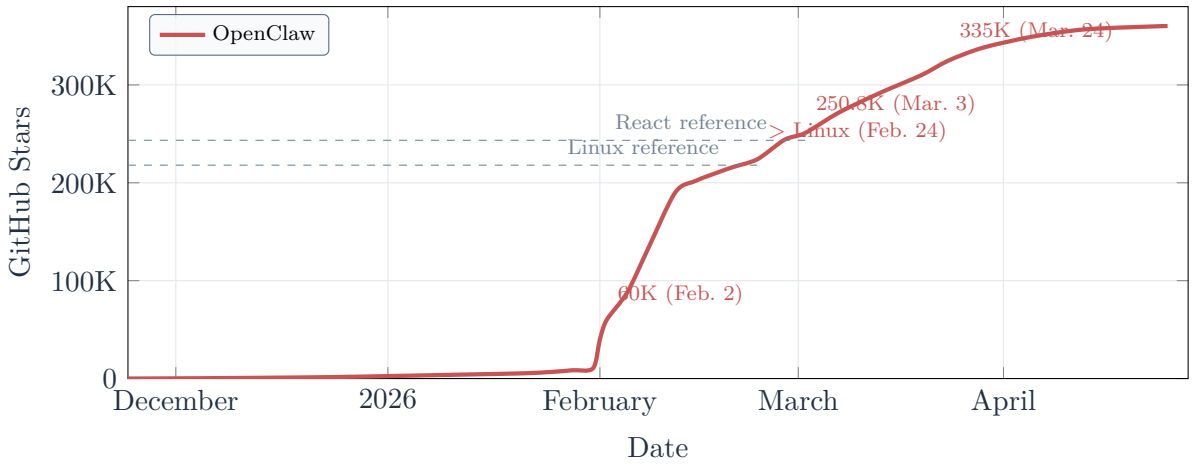


Figure 3: OpenClaw GitHub star trajectory with cross-checked milestone anchors. Dashed horizontal lines mark the Linux and React comparison levels used in the text.

adoption, and the derivative ecosystem that emerged as a response to OpenClaw's architectural constraints.

7.1 Adoption Metrics and Growth Dynamics

7.1.1 GitHub Star Growth

Figure 3 shows a near-flat pre-viral baseline through December and most of January, a sharp inflection beginning on January 29, sustained high-velocity growth through mid-February, and a later March reacceleration. In terms of the phase definitions introduced in Section 3, the initial growth wave spans Phase 1 (January 29 – February 2) and the early portion of Phase 2, while the March reacceleration corresponds to Phase 3.

The velocity comparison underscores the anomaly. OpenClaw reached 100,000 stars in 12 days; React required approximately 10 years (68 stars/day average); the Linux kernel required approximately 30 years (18 stars/day); AutoGPT, the previous fastest-growing AI project, required approximately 6 months [19][63][39]. OpenClaw's 100K-star velocity is 123 times that of React and 419 times that of Linux.

The March reacceleration is closely tied to geographic adoption dynamics. Traffic from China surged 1,436% month-over-month during March 2026, propelled by what Chinese media termed the "lobster craze" [40][39]. This China-driven growth phase was accompanied by a surge in Chinese-language tutorials and enterprise experimentation (including Tencent's ClawPro).

7.1.2 Comparative Platform Metrics

The geographic distribution of traffic reveals concentrated adoption in three regions: the United States (16.29%), India (12.16%), and China (12.08%, with month-over-month growth of 1,436%), followed by Germany (4.10%, +992% MoM) and Canada (3.53%, +1,259% MoM) [39][64].

7.1.3 Usage and Ecosystem Metrics

Beyond GitHub metrics, OpenClaw's adoption is characterized by the following indicators as of April 2026: 3M+ monthly active users; 30M+ website visitors per month;

500,000+ running instances across 80+ countries; approximately 500,000+ internet-facing instances in late-March 2026 live checks, of which 15,000+ were reported as exploitable via known RCE flaws; 40,000+ skills on ClawHub; 1,800+ repository contributors; 2M+ total skill installations; tokens processed via 350+ LLM models; 180+ ecosystem startups with combined monthly revenue exceeding $320,000; and 80+ npm packages dependent on the OpenClaw runtime [40][1][41][42][4].

7.1.4 The Adoption Paradox: Stars ≠ Enterprise Adoption

A critical finding is the disconnect between community popularity and enterprise adoption. LangGraph, with 30,000+ stars, is deployed by 400 companies in production—including Cisco, Uber, LinkedIn, BlackRock, and JPMorgan Chase—and registers 34.5 million monthly package downloads [21][52][59]. Dify, with 130,000+ stars, serves 280 enterprises with 1.4 million deployments and a $30 million fundraise [50][61]. OpenClaw, with 360,000+ stars—more than LangGraph and Dify combined—has 180+ ecosystem startups, $320,000 per month in ecosystem revenue, and zero Fortune 500 customers [40][39][1][4].

This paradox illustrates a systematic relationship: GitHub stars correlate with accessibility and viral potential, not with production maturity or enterprise trust. Enterprise adopters require security guarantees (RBAC, SSO, audit trails), operational reliability (SLAs, support contracts), and regulatory compliance—none of which OpenClaw provides. The ASTELD framework captures this relationship: OpenClaw's S2 security posture is incompatible with enterprise requirements, regardless of its D1 deployment accessibility or its community scale.

7.2 The Derivative Ecosystem

The most striking structural phenomenon in OpenClaw's ecosystem is the rapid emergence of derivative projects: over 50 forks, rewrites, and inspired-by projects appeared within three months of OpenClaw's breakout [19]. This fragmentation is not random; it is a systematic response to specific architectural constraints that the ASTELD framework makes legible.

7.2.1 Derivative Taxonomy

We classify derivatives into three tiers based on their relationship to the OpenClaw codebase and their target audience.

Tier 1: Community Derivatives. Rather than direct modifications of the OpenClaw codebase, the most prominent early derivatives were fresh implementations that re-expressed OpenClaw's core assistant model in Python, Rust, Go, and TypeScript for different priorities of simplicity, hardware efficiency, portability, and customizability [19]. NanoClaw's current fork ratio is notably anomalous at approximately 0.4+. Combined, the top four community derivatives accumulated roughly 124,500+ stars and 27,700+ forks by late April 2026 [65][66][67][68]. TrustClaw, while community-developed, specifically targets security hardening by adding OAuth and sandboxed execution (S2→S3).

Tier 2: Enterprise Adaptations. Commercial or institutional projects that extend OpenClaw for enterprise requirements. Tencent's ClawPro shifts from A1 to A2 (visual workflow) and S2 to S4 (enterprise RBAC/SSO), targeting Chinese enterprise customers. These projects uniformly shift the Security (S) and Deployment (D) axes toward enterprise-

Table 6: OpenClaw derivative ecosystem classification.

Project	Tier	Lang.	Created	Stars	Forks	ASTELD / Key Differentiation
Tier 1: Community Derivatives						
Nanobot (HKUDS)	Community	Python	2026-02-01	37.4K+	6.5K+	Reduce complexity; compact Python rewrite
ZeroClaw	Community	Rust	2026-02-13	30.6K+	4.5K+	S2→S3 (memory safety); 8.8 MB binary
PicoClaw (Sipeed)	Community	Go	2026-02-04	28.5K+	4.1K+	D1 with edge/IoT emphasis
NanoClaw	Community	TypeScript	2026-01-31	28K+	12.6K+	Simplify A1 (700 LoC); fork ratio 0.4+
TrustClaw	Community	TypeScript	2026-02	Private	Private	S2→S3 (OAuth + sandbox)
Tier 2: Enterprise Adaptations						
Tencent ClawPro	Enterprise	—	2026-03	Private	Private	A1→A2, S2→S4, D1→D2
QClaw	Enterprise	—	2026-03	Private	Private	WeChat integration; China LLM providers
MaxClaw (Minimax)	Enterprise	—	2026-03	Private	Private	China market optimization
Kimi Claw	Enterprise	—	2026-03	Private	Private	Moonshot AI integration
Tier 3: Research Platforms						
Moltworker	Research	TypeScript	2026-01	9.8K+	1.8K+	A1→A5, E1→E2 (Cloudflare sandbox deployment)
Dr. Claw	Research	TypeScript	2026-01	900+	90+	A1→A2, E1→E4 (research pipeline)

Combined Tier 1 metrics (top 4): roughly 124.5K+ stars and 27.7K+ forks within a 14-day emergence window (Jan 31 – Feb 13, 2026). Community-fork metrics were refreshed from current GitHub repository pages in late April 2026 [65][66][67][68]. Closed-source derivatives are marked as Private; the final-column notes are derived from documentation and feature analysis.

grade values, confirming that OpenClaw's S2/D1 profile is structurally incompatible with enterprise requirements.

Tier 3: Research Platforms. Projects that use OpenClaw as a foundation for academic research rather than production deployment. Moltworker shifts from A1 to A5 (multi-agent) and E1 to E2 (multi-agent conversation), enabling research on agent collaboration and delegation. Dr. Claw shifts E1 to E4 (research pipeline workflow), enabling structured experimental workflows.

7.2.2 Derivative Prediction Power

The ASTELD framework reveals that derivative shifts are non-random. Three axes—Security (S), Execution (E), and Deployment (D)—account for the vast majority of derivative innovation. No derivative has shifted the Tool Integration axis from T2 to T3/T4 (marketplace to protocol-based), suggesting that the marketplace model is not perceived as the primary constraint by derivative developers, despite its role in ClawHavoc. Similarly, no derivative has shifted the Architecture axis from A1 to A4 (monolithic to graph-based), suggesting that the monolithic daemon pattern is perceived as a feature, not a bug, by the community.

This pattern has predictive implications. Future derivatives are most likely to target

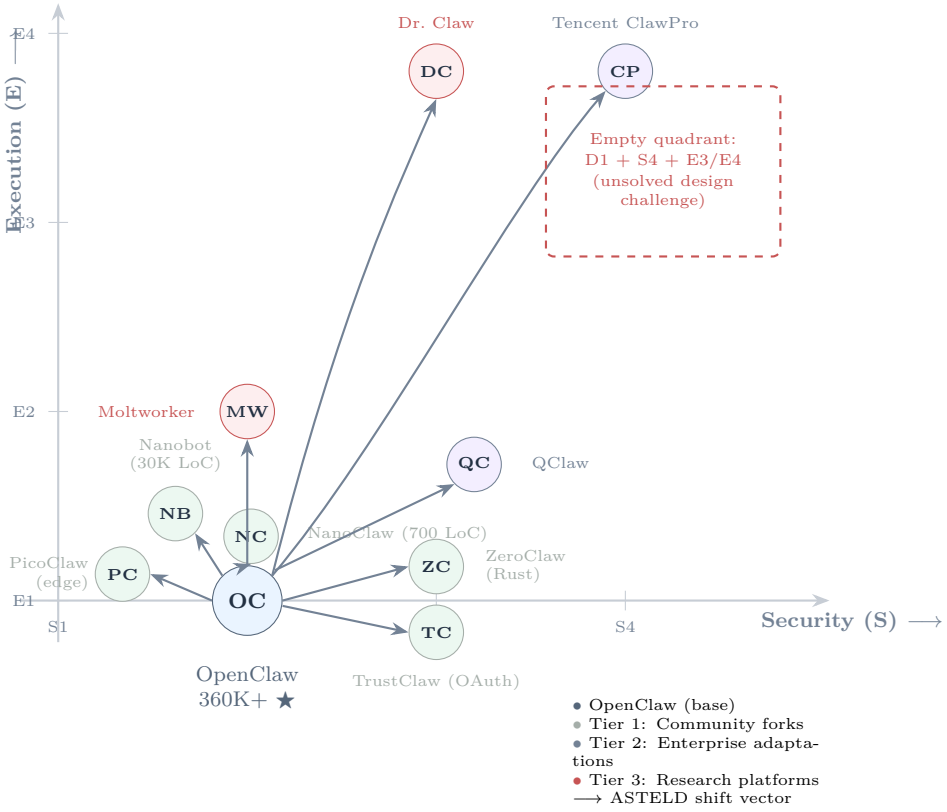


Figure 4: OpenClaw derivative ecosystem on the ASTELD Security–Execution plane. Arrows indicate the shift from OpenClaw's base position (S2, E1).

the Security-Deployment gap identified in Section 6: a local-first agent with enterprise-grade security. The current empty quadrant (D1 + S4) represents the most commercially valuable unsolved design challenge in the autonomous agent space.

7.3 Social and Regulatory Impact

7.3.1 The Token Economy

OpenClaw has catalyzed a nascent token economy. As of April 2026, OpenClaw instances are estimated to have collectively processed trillions of tokens across 350+ language models [40]. The 180+ ecosystem startups—skill developers, hosting providers, integration services—generate combined monthly revenue exceeding $320,000 [40][39][1]. While modest relative to the broader AI industry ($7.84 billion global AI agent market in 2025, projected to reach $52.62 billion by 2030 at 46.3% CAGR [39][42]), this ecosystem represents the first organic token-mediated economy around an open-source AI agent.

7.3.2 ClawHub Marketplace Dynamics

The ClawHub marketplace has grown to 40,000+ skills. However, the security profile is concerning: independent assessment finds 47% of skills assessed as safe, 36% containing prompt injection vectors, 8% attempting data exfiltration, and 6% requesting excessive permissions [40]. The trend is worsening: at the time of the ClawHavoc disclosure in February 2026, approximately 12% of the then-10,700 skills were classified as malicious [7]; by April, 36% of the now-40,000+ skills contain prompt injection [40][1].

Because these two assessments use different classification schemes—broader unsafe or adversarial characteristics in the former case, versus explicitly malicious skills in the latter—they are not directly comparable as a single time-series. Even so, taken together, they indicate that marketplace risk is worsening faster than marketplace governance is maturing. Therefore, the governance challenge—enabling permissionless contribution while preventing compromise—remains the central unsolved problem for agent skill ecosystems (see RQ5, Section 8).

7.3.3 Regulatory Responses

OpenClaw has begun to draw governmental attention to autonomous AI agents, though no formal multi-jurisdictional review process has yet been publicly confirmed.

The broader regulatory landscape remains uncertain. The EU AI Act classifies AI systems by risk level but does not specifically address autonomous agents with system-level access. The NIST AI Risk Management Framework provides general principles but no agent-specific guidance. No academic analysis has mapped how existing regulatory frameworks apply to the specific capabilities and risks of autonomous AI agents (see RQ7, Section 8).

7.3.4 Enterprise Adoption Dynamics

The enterprise adoption paradox identified in Section 7.1.4 reflects a broader market dynamic. Gartner projects that 40% of enterprise applications will incorporate AI agents by end of 2026 (up from less than 5% in 2025), and 80% of Fortune 500 companies are exploring agent deployments [39]. Yet this adoption is overwhelmingly concentrated in enterprise-grade frameworks: LangGraph (400 production deployments among named enterprises), Dify (280 enterprises, $30M fundraise), and Microsoft Agent Framework (Azure integration, Semantic Kernel ecosystem) [21][50][51].

OpenClaw's path to enterprise adoption is blocked by structural factors captured in its ASTELD profile: S2 security posture (no RBAC, no SSO, no audit trails), D1 deployment topology (no server-grade deployment option by default), and the ongoing security advisory accumulation (averaging more than one per day) [5]. The derivative ecosystem's enterprise tier (Tencent ClawPro, QClaw) represents market attempts to bridge this gap, but none has yet achieved the scale of LangGraph or Dify in enterprise settings.

7.4 Summary

OpenClaw's adoption metrics are record-setting by every community measure: stars, growth velocity, monthly active users, and ecosystem breadth. Yet the adoption paradox—360,000+ stars but zero Fortune 500 customers—reveals that community scale and enterprise readiness are orthogonal dimensions. The derivative ecosystem, with 50+ projects appearing in three months, is not fragmentation for its own sake but a systematic market response to specific ASTELD-axis constraints. The three most constrained axes—Security, Execution, and Deployment—predict the direction of derivative innovation with remarkable accuracy. The social and regulatory impacts—the token economy, ClawHub's worsening security profile, emerging regulatory attention, and the enterprise adoption gap—point to fundamental open questions that we formulate in the next section.

8 Open Research Questions and Future Directions

The preceding analysis reveals a field in rapid but uneven maturation. The technical capability of autonomous AI agents has advanced ahead of security infrastructure, governance frameworks, and academic analysis. This section formulates eight research questions grounded in the evidence gaps identified throughout this survey, organized by research domain.

8.1 Architectural Research

RQ1: Can we design a formally verifiable architecture that enforces a strict separation between instruction channels and data channels in autonomous AI agents, thereby eliminating prompt injection as an authorization vulnerability?

As shown in Section 4, OpenClaw adopts a monolithic architecture in which the LLM controller processes both user instructions and untrusted external data within a shared context window, without any privilege separation between reasoning and execution components. Section 5 further demonstrates that this design directly enables multiple classes of prompt injection attacks, including indirect injection, guidance injection, and memory poisoning, all of which exploit the absence of a structural boundary between instruction and data processing.

These findings collectively suggest that prompt injection is not an isolated vulnerability, but a direct consequence of architectural design. Existing approaches, such as Energent.ai's multi-step parsing, attempt partial separation but lack formal verification or formal security guarantees [45]. This motivates the need for a provably secure architectural paradigm that enforces instruction–data separation at the system level. The relevant formalism may draw on information flow control, capability-based security, or language-theoretic security.

RQ2: What is the principled design of a capability-based permission model for AI agent tool ecosystems that balances security, usability, and composability?

As shown in Section 4, OpenClaw's tool execution engine operates with the full privileges of the user account, without any built-in mechanism for permission scoping or isolation. Section 5 further reveals that this unrestricted access is a key enabler of multiple attack vectors, particularly supply chain attacks (Category E) and prompt injection escalations (Category D), where malicious skills or injected instructions can trigger arbitrary system-level actions.

The ClawHavoc campaign provides empirical evidence that the absence of a permission model allows large-scale compromise through seemingly benign skill installations. More broadly, these findings indicate that the current implicit full-trust model is fundamentally incompatible with secure agent deployment.

CertiK accordingly recommended that skills should declare resource needs upfront, analogous to Android's permission model [16]. However, no existing framework implements such a mechanism at the architectural level. The design space includes static declarations at install time, dynamic capability negotiation at runtime, and sandboxed execution with permission escalation. A central challenge is to determine the appropriate permission granularity and enforcement model that can preserve open-source contribution velocity while reducing the risk of systemic compromise.

8.2 Security Research

RQ3: How can the correctness of human-in-the-loop approval mechanisms in autonomous agents be formally defined and evaluated, given the observed gap between approval-time representations and execution-time behavior?

As shown in Section 5, OpenClaw's approval system has been bypassed through multiple CVEs, each exploiting inconsistencies between the approval-time display of a command and its execution-time resolution. These cases reveal a systematic misalignment between what users approve and what the system ultimately executes.

This problem is analogous to time-of-check-to-time-of-use (TOCTOU) vulnerabilities, but arises at a higher abstraction level where the approved entity is a natural language or semi-structured representation of an action, rather than a fully specified program. A key open challenge is to define what constitutes "approval correctness" in such systems and to determine whether it can be formally specified, verified, or empirically evaluated under realistic agent behavior.

RQ4: How can AI agent supply chain attacks be detected and mitigated when malicious behavior is encoded in natural language rather than executable code?

Section 5 demonstrates that agent supply chain attacks, such as the ClawHavoc campaign, operate through natural language payloads embedded in skill descriptions and execution flows, rather than traditional executable artifacts. As a result, conventional static analysis tools, including VirusTotal, are ineffective in identifying such threats [8][33].

These observations suggest a shift in the nature of supply chain risk, where the LLM itself acts as the interpreter of potentially malicious logic. This raises a fundamental challenge: how to identify and constrain harmful behavior when it is represented as semantically meaningful but syntactically benign text.

An open question is what forms of analysis—semantic, behavioral, or provenance-based—are sufficient to detect such attacks, and how these approaches can scale with the rapid growth of open agent ecosystems.

8.3 Ecosystem Research

RQ5: What governance models can sustain open-source AI agent ecosystems while mitigating systemic supply chain risks at scale?

As discussed in Section 6, OpenClaw's plugin-based ecosystem enabled rapid growth but also expanded the attack surface, while its permissive licensing and lack of built-in governance mechanisms provided no structural safeguards against malicious contributions. The ClawHavoc campaign illustrates how such a largely permissionless contribution model can lead to large-scale compromise in agent skill marketplaces.

In contrast, tightly controlled ecosystems such as Apple's App Store significantly reduce supply chain risk through centralized review, but at the cost of reduced openness and developer flexibility. This contrast highlights a fundamental tension between accessibility and security in agent ecosystems.

While intermediate approaches—such as reputation systems, staged trust models, or automated screening—have been explored in other software ecosystems, their applicability and effectiveness in AI agent ecosystems remain unclear. An open question is how governance structures can be designed and evaluated to balance ecosystem growth, developer participation, and systemic security under adversarial conditions.

RQ6: How do the architectural choices of single-agent (OpenClaw) versus multi-agent (CrewAI/AutoGen) systems affect fault isolation, security boundary enforcement, and emergent behavior at deployment scale?

As discussed in Section 6, the distinction between single-agent and multi-agent architectures is one of the most fundamental design choices in the agent framework landscape, yet their comparative security and reliability properties remain poorly understood. The Moltbook phenomenon—emergent collaborative behaviors among multiple OpenClaw instances communicating through social networks [13][12]—suggests that multi-agent dynamics can arise even in systems not explicitly designed for them, with unclear implications for coordination, fault isolation, and security boundaries. In addition, recent evidence of cross-agent prompt injection propagation indicates that vulnerabilities may spread across interacting agents, but no formal threat model yet exists for such behaviors.

This raises an open question of how architectural choices across the single-agent/multi-agent spectrum shape blast radius of compromise, boundary enforcement, emergent coordination, and performance trade-offs in real-world deployments.

8.4 Sociotechnical Research

RQ7: What factors explain the divergent regulatory responses to autonomous AI agents across jurisdictions, and how do existing AI governance frameworks apply to agentic systems?

Recent developments suggest growing regulatory attention to autonomous AI agents across multiple jurisdictions, including emerging scrutiny of agent-level system access and data governance concerns. However, as discussed in Section 7, there is limited systematic analysis of how these emerging responses relate to existing AI governance frameworks.

While frameworks such as the EU AI Act and the NIST AI Risk Management Framework provide general approaches to risk classification and lifecycle governance, it remains unclear how well they capture the specific properties of agentic systems, including system-level access, persistent state, and autonomous multi-step action.

This raises an open question of how existing regulatory frameworks can be interpreted, adapted, or extended to address agentic AI, and what factors drive variation in regulatory responses across jurisdictions.

RQ8: How can the security risk of autonomous AI agents be systematically characterized and quantified for enterprise adoption decisions, given the absence of standardized benchmarks and the rapid evolution of the threat landscape?

As discussed in Section 7, there is a notable gap between community adoption and enterprise deployment, with highly popular agent platforms lacking corresponding enterprise uptake. This discrepancy is partly attributed to the absence of standardized methods for assessing security and operational risk in agent systems.

Existing efforts, such as the Personalized Agent Security Bench (PASB) [43], provide initial evaluation directions, but remain limited in scope and adoption. In practice, enterprise decision-making relies on heterogeneous sources, including vendor claims, security audits, and informal assessments [15][16], resulting in inconsistent and difficult-to-compare evaluations.

These observations suggest an open question of how risk in agent systems can be defined, measured, and compared, particularly given their unique characteristics such as persistent state, tool access, natural language attack surfaces, and multi-step execution

under dynamic conditions.

8.5 Meta-Observations

Three cross-cutting observations emerge from the research questions above:

Security failures originate from missing boundaries across abstraction levels. The challenges identified in RQ1–RQ4 collectively point to a common issue: the absence of clear separation between instruction, data, and execution across the agent stack. Whether at the architectural level (RQ1), human-in-the-loop control (RQ3), or ecosystem interfaces (RQ4–RQ5), vulnerabilities arise when control signals and operational effects are not cleanly aligned. This suggests that agent security is fundamentally a problem of boundary definition and enforcement, rather than isolated implementation flaws.

Irreducible trade-offs drive ecosystem stratification. The questions raised in RQ5 and RQ6 reflect a deeper tension between openness, flexibility, and security. Empirical evidence from the ecosystem analysis and derivative landscape shows that no single architecture simultaneously satisfies these objectives, leading to the emergence of specialized system tiers. Rather than convergence, the field is undergoing structured fragmentation, implying that future progress depends on interoperability and cross-system guarantees rather than universal design solutions.

Adoption is constrained by unresolved sociotechnical alignment. The issues highlighted in RQ7 and RQ8 indicate that technical capability alone is insufficient for widespread deployment. Gaps in governance models, regulatory interpretation, and risk quantification create barriers to enterprise and institutional adoption. This suggests that the long-term trajectory of agent systems will depend on aligning technical design with policy frameworks, organizational practices, and economic incentives.

9 Conclusion

This paper has presented the first comprehensive, multi-dimensional survey of OpenClaw and the autonomous AI agent landscape. Over the course of five dedicated analytical dimensions—architecture, security, ecosystem comparison, derivative ecosystem, and adoption dynamics—with governance examined as a cross-cutting theme, we have documented a technology that embodies both the transformative potential and the systemic risks of the agent paradigm.

Our five contributions are as follows. First, we provided the most extensive multi-dimensional survey of OpenClaw to date, covering five interrelated dimensions. Second, we developed a six-category security vulnerability taxonomy with cross-source triangulation, synthesizing evidence from CVE databases, five institutional assessments, and academic research—including the first comprehensive account of the ClawHavoc supply chain campaign that led to the identification of 1,184 malicious marketplace skills. Third, we introduced the ASTELD classification framework, a six-axis taxonomy for autonomous AI agent platforms that is discriminative, empirically grounded, and extensible, yielding distinct mapped profiles for the eight surveyed frameworks under dominant configurations. Fourth, we mapped the derivative ecosystem, classifying 50+ projects across three tiers and demonstrating that the ASTELD framework can predict the direction of ecosystem fragmentation: derivative innovation concentrates on the three most constrained ASTELD axes (Security, Execution, Deployment). Fifth, we formulated eight actionable research

questions spanning architectural, security, ecosystem, and sociotechnical domains, each grounded in identified evidence gaps.

The central thesis of this survey is that OpenClaw's trajectory—from weekend project to the most-starred repository in GitHub history to a case study in security debt—is not an anomaly but a crystallization of structural tensions inherent to the autonomous AI agent paradigm. The accessibility-security trade-off, the open-source speed paradox, and the impossibility of a single architecture satisfying all use cases are not problems specific to OpenClaw; they are properties of the design space itself.

Three findings carry particular weight. The security-accessibility diagonal identified through the ASTELD framework reveals that none of the surveyed platforms combines local-first deployment accessibility with enterprise-grade security—an empty quadrant that represents the most consequential unsolved design challenge in the field. The adoption paradox—360,000+ GitHub stars but zero Fortune 500 customers—demonstrates that community popularity and enterprise readiness are orthogonal metrics, mediated by security posture and governance maturity rather than community scale. The worsening ClawHub marketplace profile—from 12% malicious skills in February to 36% containing prompt injection vectors by April across a 40,000+ skill marketplace—suggests that reactive defenses cannot maintain marketplace integrity at scale, and that the governance of agent extension ecosystems requires fundamentally new approaches.

The autonomous AI agent paradigm is real, consequential, and here to stay. The question is no longer whether agents will be deployed at scale but whether the research community, the open-source ecosystem, and regulatory bodies can close the gap between capability and safety before the next ClawHavoc.

References

[1] OpenClawVPS. OpenClaw statistics 2026: Growth, users, security, and data, 2026. URL https://openclawvps.io/blog/openclaw-statistics. Public metric roundup citing GitHub, Similarweb, npm, and security reporting.

[2] Aftab. OpenClaw just beat React's 10-year GitHub record in 60 days, 2026. URL https://medium.com/@aftab001x/openclaw-just-beat-reacts-10-year-github-record-in-60-days-now-nobody-knows-what-to-do-with-it-937b8f370507.

[3] OpenClaw Blog. 250,000 stars: OpenClaw surpasses React, 2026. URL https://openclaws.io/blog/openclaw-250k-stars-milestone.

[4] GitHub. openclaw/openclaw, 2026. URL https://github.com/openclaw/openclaw. Repository page accessed April 25, 2026.

[5] jgamblin. OpenClawCVEs tracker, 2026. URL https://github.com/jgamblin/OpenClawCVEs/. GitHub repository tracking all OpenClaw CVEs and advisories.

[6] Blink. Openclaw security overview: 138 cves and severity distribution, April 2026. URL https://www.blinkops.com/blog/openclaw-security-overview-138-cves-risk-and-response. Vendor security overview aggregating 138 tracked vulnerability records and severity counts; broader than the manuscript's narrower published-CVE count.

[7] CyberPress. ClawHavoc poisons OpenClaw's ClawHub with 1,184 malicious skills, 2026. URL https://cyberpress.org/clawhavoc-poisons-openclaws-clawhub-with-1184-malicious-skills/.

[8] Koi Security. ClawHavoc: 341 malicious clawed skills found by the bot they were

targeting, February 2026. URL https://www.koi.ai/blog/clawhavoc-341-malicious-clawedbot-skills-found-by-the-bot-they-were-targeting.

[9] Surada Suwansathit, Yuxuan Zhang, and Guofei Gu. A systematic taxonomy of security vulnerabilities in the OpenClaw ai agent framework. *arXiv preprint arXiv:2603.27517*, 2026.

[10] Zhengyang Shan, Jiayun Xin, Yue Zhang, and Minghui Xu. Don't let the claw grip your hand: A security analysis and defense framework for OpenClaw. *arXiv preprint arXiv:2603.10387*, 2026.

[11] Preprints.org. OpenClaw as language infrastructure: A case-centered survey of a public agent ecosystem in the wild. *Preprints.org*, 2026. NLP-centered survey using GATE/AERO frameworks on 38 papers.

[12] Md Motaleb Hossen Manik and Ge Wang. OpenClaw agents on Moltbook: Risky instruction sharing and norm enforcement in an agent-only social network. *arXiv preprint arXiv:2602.02625*, 2026.

[13] OpenClaw. Agent runtime — OpenClaw official documentation, 2026. URL https://docs.openclaw.ai/concepts/agent.

[14] SSRN. OpenClaw and the anatomy of contemporary ai hype: A methodological audit and the case for large understanding models (LUM). *SSRN*, 2026. SSRN:6335019. Critical perspective on AI agent hype cycle.

[15] Microsoft Defender Security Research Team. Running OpenClaw safely: Identity, isolation, and runtime risk, February 2026. URL https://www.microsoft.com/en-us/security/blog/2026/02/19/running-openclaw-safely-identity-isolation-runtime-risk/. Microsoft Security Blog.

[16] CertiK. OpenClaw security report, March 2026. URL https://www.certik.com/blog/openclaw-security-report.

[17] Elia Zaitsev. What security teams need to know about OpenClaw, the ai super agent, February 2026. URL https://www.crowdstrike.com/en-us/blog/what-security-teams-need-to-know-about-openclaw-ai-super-agent/. CrowdStrike Blog.

[18] Fernando Tucci. CISOs in a pinch: A security analysis of OpenClaw, March 2026. URL https://www.trendmicro.com/en_us/research/26/c/cisos-in-a-pinch-a-security-analysis-openclaw.html. Trend Micro Research.

[19] OSS Insight. 116,000 stars in 8 weeks: Four teams rewrote OpenClaw, 2026. URL https://ossinsight.io/blog/the-openclaw-forks-wave-2026.

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[21] LangChain Blog. LangChain and LangGraph agent frameworks reach v1.0 milestones, 2025. URL https://blog.langchain.com/langchain-langgraph-1dot0/.

[22] GitHub. Significant-Gravitas/AutoGPT, 2026. URL https://github.com/Significant-Gravitas/AutoGPT. Repository page accessed in late April 2026.

[23] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Proceedings of the First Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS. Conference version of the AutoGen framework paper.

[24] The Agent Times. CrewAI: Multi-agent collaboration, 2026. URL https://theagenttimes.com/articles/44335-stars-and-counting-crewais-github-surge-maps-the-rise-of-the-multi-agent-e.

[25] Milvus Blog. What is OpenClaw? complete guide to the open-source ai agent, 2026. URL https://milvus.io/blog/openclaw-formerly-clawdbot-moltbot-explained-a-complete-guide-to-the-autonomous-ai-agent.md.

[26] KDnuggets. OpenClaw explained: The free ai agent tool going viral already in 2026, 2026. URL https://www.kdnuggets.com/openclaw-explained-the-free-ai-agent-tool-going-viral-already-in-2026.

[27] Arunkumar V, Gangadharan G.R., and Rajkumar Buyya. Agentic artificial intelligence (AI): Architectures, taxonomies, and evaluation of large language model agents. *arXiv preprint arXiv:2601.12560*, 2026.

[28] Bin Xu. AI agent systems: Architectures, applications, and evaluation. *arXiv preprint arXiv:2601.01743*, 2026.

[29] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18:186345, 2024. doi: 10.1007/s11704-024-40231-1. URL https://link.springer.com/article/10.1007/s11704-024-40231-1.

[30] Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. AI agents vs. Agentic AI: A conceptual taxonomy, applications and challenges. *Information Fusion*, 2025.

[31] Interface EU. An autonomy-based classification of ai agents. In *NeurIPS 2024 Workshop*, 2024. 5-level autonomy classification for policy and liability.

[32] Penligent AI. The OpenClaw prompt injection problem: Persistence, tool hijack, 2026. URL https://www.penligent.ai/hackinglabs/the-openclaw-prompt-injection-problem-persistence-tool-hijack-and-the-security-boundary-that-doesnt-exist/.

[33] Arnav. Why you cannot prevent prompt injection, 2026. URL https://arnav.au/2026/04/02/why-you-cannot-prevent-prompt-injection/.

[34] Oasis Security Research Team. ClawJacked: OpenClaw vulnerability enables full agent takeover, February 2026. URL https://www.oasis.security/blog/openclaw-vulnerability.

[35] ARMO. CVE-2026-32922: Critical privilege escalation in OpenClaw, 2026. URL https://www.armosec.io/blog/cve-2026-32922-openclaw-privilege-escalation-cloud-security/.

[36] Wikipedia contributors. OpenClaw — Wikipedia, 2026. URL https://en.wikipedia.org/wiki/OpenClaw. Accessed April 2026.

[37] Peter Steinberger. OpenClaw, OpenAI and the future, 2026. URL https://steipete.me/posts/2026/openclaw.

[38] OpenClaw Blog. OpenClaw creator peter steinberger joins OpenAI, 2026. URL https://openclaws.io/blog/openclaw-creator-joins-openai/.

[39] star-history.com. OpenClaw star history, 2026. URL https://www.star-history.com/openclaw/openclaw. Interactive GitHub star growth visualization.

[40] Finn Hillebrandt and Gradually.ai. OpenClaw statistics 2026: Key numbers, data & facts, 2026. URL https://www.gradually.ai/en/openclaw-statistics/. Statistics roundup citing Similarweb, ClawHub, GitHub, and OpenRouter.

[41] SecurityScorecard. How exposed openclaw deployments turn agentic ai into an attack surface, February 2026. URL https://securityscorecard.com/blog/how-exposed-openclaw-deployments-turn-agentic-ai-into-an-attack-surface/. STRIKE research on exposed OpenClaw instances and RCE exposure rate.

[42] Louis Columbus. Openclaw has 500,000 instances and no enterprise kill switch, March 2026. URL https://venturebeat.com/security/openclaw-500000-instances-no-enterprise-kill-switch/. VentureBeat report citing an RSAC 2026 interview with Cato Networks.

[43] Yuhang Wang, Feiming Xu, Zheng Lin, Guangyu He, Yuzhe Huang, Haichang Gao, Zhenxing Niu, Shiguo Lian, and Zhaoxiang Liu. From assistant to double agent: Formalizing and benchmarking attacks on OpenClaw for personalized local ai agent. *arXiv preprint arXiv:2602.08412*, 2026. Proposes the Personalized Agent Security Bench (PASB).

[44] The Hacker News. ClawJacked flaw lets malicious sites hijack local OpenClaw ai agents, 2026. URL https://thehackernews.com/2026/02/clawjacked-flaw-lets-malicious-sites.html.

[45] eSecurity Planet. OpenClaw or open door? prompt injection creates ai backdoors, 2026. URL https://www.esecurityplanet.com/threats/openclaw-or-open-door-prompt-injection-creates-ai-backdoors/.

[46] Repello AI. ClawHavoc: Inside the supply chain attack that targeted 300,000 ai agent users, 2026. URL https://repello.ai/blog/clawhavoc-supply-chain-attack.

[47] OpenClaw Pulse. OpenClaw security guide 2026: Every CVE, exploit and fix, 2026. URL https://openclawpulse.com/openclaw-security-guide-2026/.

[48] GitHub. NousResearch/hermes-agent, 2026. URL https://github.com/NousResearch/hermes-agent. Repository page accessed April 26, 2026.

[49] n8n. n8n: Ai workflow automation platform, 2026. URL https://n8n.io/.

[50] Dify Blog. Dify: 100k stars on GitHub, 2025. URL https://dify.ai/blog/100k-stars-on-github-thank-you-to-our-amazing-open-source-community.

[51] Microsoft Learn. Microsoft agent framework overview, 2026. URL https://learn.microsoft.com/en-us/agent-framework/overview/agent-framework-overview.

[52] Let's Data Science. Ai agent frameworks 2026: LangGraph vs CrewAI & more, 2026. URL https://letsdatascience.com/blog/ai-agent-frameworks-compared.

[53] openclaw-ai.net. OpenClaw vs AutoGPT: Which ai agent framework should you choose, 2026. URL https://openclaw-ai.net/en/blog/openclaw-vs-autogpt.

[54] NocoBase. Top 20 ai projects on GitHub 2026: Not just OpenClaw, 2026. URL https://www.nocobase.com/en/blog/best-open-source-ai-projects-github-2026.

[55] Hassan Abou Ali et al. Agentic AI: A comprehensive survey of architectures, applications, and future directions. *Artificial Intelligence Review*, 2025. Dual-paradigm framework (symbolic vs neural); PRISMA review of 90 studies.

[56] Francesco Piccialli et al. AgentAI: A comprehensive survey on autonomous agents in distributed ai for industry 4.0. *Expert Systems with Applications*, 2025.

[57] Frontiers in Computer Science. The dark side of autonomous intelligence: Data leakage and privacy failures in agentic AI. *Frontiers in Computer Science*, 2026. 5-category data leakage taxonomy (memory, tools, planning, inter-agent, feedback).

[58] GitHub. crewAIInc/crewAI, 2026. URL https://github.com/crewAIInc/crewAI. Repository page accessed April 25, 2026.

[59] GitHub. langchain-ai/langgraph, 2026. URL https://github.com/langchain-ai/langgraph. Repository page accessed April 25, 2026.

[60] GitHub. n8n-io/n8n, 2026. URL https://github.com/n8n-io/n8n. Repository page accessed April 25, 2026.

[61] GitHub. langgenius/dify, 2026. URL https://github.com/langgenius/dify. Repository page accessed April 25, 2026.

[62] GitHub. microsoft/autogen, 2026. URL https://github.com/microsoft/autogen. Repository page accessed in late April 2026.

[63] star-history.com. Star history monthly september 2024 | ai agents, 2024. URL https://www.star-history.com/blog/ai-agents/. Blog post published on September 19, 2024.

[64] fatjoe. OpenClaw AI Stats 2026: Uses, Users, Market Share, and More, 2026. URL https://fatjoe.com/blog/openclaw-ai-stats/. Statistics roundup published on March 11, 2026; includes country traffic share and month-over-month change figures, citing Similarweb for web traffic data.

[65] GitHub. HKUDS/NanoBot, 2026. URL https://github.com/HKUDS/NanoBot. Repository page accessed April 25, 2026.

[66] GitHub. zeroclaw-labs/zeroclaw, 2026. URL https://github.com/zeroclaw-labs/zeroclaw. Repository page accessed April 25, 2026.

[67] GitHub. sipeed/picoclaw, 2026. URL https://github.com/sipeed/picoclaw. Repository page accessed April 25, 2026.

[68] GitHub. qwibitai/nanoclaw, 2026. URL https://github.com/qwibitai/nanoclaw. Repository page accessed April 25, 2026.