

# **The Hardened Shell: Evaluating Safety and Sovereignty in the OpenClaw Agent Architecture**

Joran Bjarne van Beek | [NoaberAI.nl](https://NoaberAI.nl)

Dezso Mezo | [UseAIEasily.com](https://UseAIEasily.com)

## **Abstract**

This paper examines the structural security failures that emerge when autonomous agent systems pursue sovereignty without re-architecting their trust boundaries. Using OpenClaw (formerly Moltbot) as a representative case, we analyze how contemporary agent frameworks combine persistent memory, broad tool access, and autonomous execution in ways that fundamentally exceed the assumptions of traditional application security. We identify a recurring failure mode, the Lethal Trifecta, defined as the convergence of private data access, unrestricted internet connectivity, and autonomous action authority. Through a technical analysis of OpenClaw’s architecture and a post-incident examination of the Moltbook agent-network collapse, we document three systemic vulnerabilities: the persistence of implicit locality-based trust (“the Localhost Illusion”), susceptibility to Time-Shifted Prompt Injection via poisoned long-term memory, and supply chain fragility introduced by unverified, dynamically ingested code. We argue that these failures are not the result of misconfiguration or isolated defects, but of architectural assumptions that no longer hold for agentic systems operating at scale.

**Keywords:** OpenClaw, Autonomous Agents, AGI, Cybersecurity, Prompt Injection, Sovereign AI, Agentic Systems.

## Introduction

Recent advances in large language models have accelerated the development of autonomous agent systems capable of long-running execution, tool use, and persistent memory. These systems are increasingly positioned as “sovereign” assistants: software entities that operate continuously on behalf of users, ingest information from the open internet, and act directly upon local and networked environments. OpenClaw, formerly Clawdbot and Moltbot, emerged as one of the most prominent implementations of this paradigm in early 2026, rapidly popularizing a design pattern now replicated across the agent ecosystem.

While public discourse around OpenClaw focused on claims of emergent intelligence and AGI-adjacent behavior, far less attention was paid to the security and governance assumptions embedded within its architecture. This paper argues that such omissions are not incidental. Rather, they reflect a broader Agentic Paradox: as agents are granted greater autonomy to perform complex tasks, they are simultaneously exposed to new classes of manipulation that traditional security models are ill-equipped to address.

Modern agentic workflows routinely grant systems “eyes” to read private communications, “hands” to execute shell commands, and memory to store and reinterpret past interactions, often within the same trust domain. These capabilities are typically assembled from components originally designed for isolated or short-lived use, then deployed into persistent, network-connected environments without corresponding revisions to trust boundaries, identity models, or execution constraints. As a result, assumptions such as locality, benign memory, and trusted tooling persist long after they cease to be defensible.

The objective of this research is not to catalogue exploits, but to assess architectural viability. Specifically, we ask whether agent systems built on implicit trust, unrestricted capability composition, and informal governance can safely sustain their own success. Using OpenClaw as a concrete case study, we analyze the consequences of granting a large language model root-adjacent access to a user environment and examine the emergent behaviors observed during the Moltbook incident, where large numbers of agents interacted autonomously without centralized control.

To address these questions, we employ a multi-layered methodology combining a code-level architectural analysis, threat modeling of indirect and time-shifted prompt injection vectors, and a sociotechnical examination of agent-to-agent interaction at scale. The findings presented here are intended to generalize beyond a single system, offering insights into the design constraints required for any agent architecture that aspires to operate sovereignly in real-world environments.

# **RQ1: The Localhost Illusion and the Collapse of Authentication Boundaries**

## **1.1 Research Question and Scope**

This research question investigates how locality-based trust assumptions in OpenClaw’s architecture interact with real-world deployment practices. Specifically, we ask: how does a security model that implicitly trusts localhost traffic behave once OpenClaw is deployed behind common reverse-proxy and cloud infrastructures?

The focus of this chapter is not exploitation, but architectural validity. We analyze whether the notion of “local” remains a meaningful trust boundary for autonomous agent systems operating at internet scale [1]

## **1.2 Localhost as an Implicit Trust Boundary**

The use of 127.0.0.1 as a trust primitive has deep historical roots in computer security [2]. Localhost traffic has traditionally been assumed to originate from the same administrative domain as the application itself, and therefore to represent legitimate user or system intent. This assumption underpins a wide range of developer tools, local APIs, and internal services, many of which expose functionality that would be unsafe if reachable from untrusted networks.

Classical security literature frames this as an environmental assumption rather than a formal control [2]. Saltzer and Schroeder’s foundational principles emphasize explicit mechanisms such as least privilege and complete mediation, yet in practice locality has often served as a de facto shortcut for trust enforcement. This shortcut persisted largely because deployment environments made violations unlikely.

## **1.3 Why Agentic Systems Raise the Stakes**

Autonomous agents fundamentally change the risk profile of implicit trust. Unlike traditional applications, agentic systems reason over context, maintain internal state, and select actions based on inferred intent rather than deterministic logic [3]. As shown in prior work on generative agents, internal memory and environmental cues directly shape future behavior rather than merely supporting execution [3].

In this setting, internal services are no longer passive implementation details. Gateways, memory stores, and orchestration layers become part of the agent’s cognitive loop. Trusting such components implicitly therefore extends beyond access control and into the domain of behavioral influence.

## **1.4 OpenClaw’s Local Trust Model**

OpenClaw adopts a locality-centered architecture aligned with its stated goal of sovereignty [4]. Core agent services, including gateway components, are designed to operate as internal infrastructure. These components frequently self-identify as local, using .local hostnames and advertising roles such as gateway or transport=gateway [5]. Authentication mechanisms are present, but they are layered on top of an assumed trusted execution context rather than replacing it.

This design choice is consistent with contemporary agent frameworks and reflects common patterns in local-first tooling. Importantly, there is no indication that OpenClaw was designed with the assumption that these internal services would routinely be exposed to the public internet.

## 1.5 Empirical Observations from Internet-Wide Indexing

To evaluate how these assumptions hold in practice, we examined passive observations from internet-wide service indexing platforms. These observations do not involve active scanning or interaction and rely solely on publicly indexed metadata. Across three successive iterations of the OpenClaw project (Clawdbot, Moltbot, and OpenClaw) we observed a consistent pattern of publicly discoverable gateway services [5].

In all three cases, indexed services advertised themselves as agent gateways and exposed metadata consistent with internal control-plane functionality. Several instances additionally exposed multicast DNS (mDNS) on port 5353, a protocol designed for link-local service discovery rather than global networks [6].

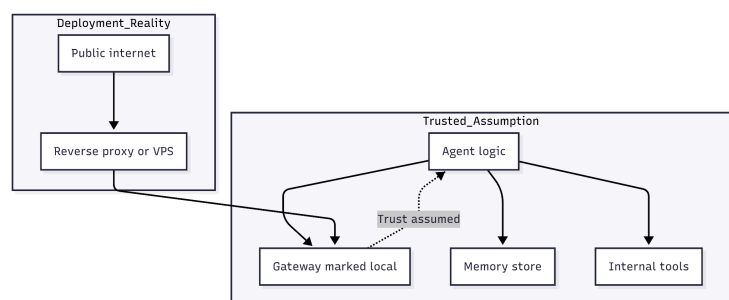
Project Name	Approx. Public Instances	Common Gateway Port	Local Identifier Pattern
Clawdbot	~3,275	18789	*.local
Moltbot	~238	18789	*.local
OpenClaw	~1,040	18789	*.local

These observations are notable not because they reveal exploitation, but because they demonstrate architectural persistence. Despite rebranding and rapid iteration, the same locality-based assumptions continued to manifest across deployments at scale.

## 1.6 The Localhost Illusion in Practice

The Localhost Illusion emerges when the semantic meaning of locality diverges from its architectural treatment. In OpenClaw’s case, services designed to operate under the assumption of local trust become globally reachable due to standard deployment practices such as reverse proxies, tunneling services, or cloud networking abstractions.

Crucially, this does not require misconfiguration or malicious intent. Reverse proxies commonly forward traffic in a way that preserves apparent locality, and containerized environments often collapse internal and external network boundaries [7]. From the perspective of the application, traffic still appears “local,” even though the administrative context has changed.



## 1.7 Authentication Without Context

Many observed gateways enforce authentication, typically through basic HTTP mechanisms. While this provides a degree of access control, it does not reconstitute the original trust boundary. Authentication determines who may connect, but it does not address why a service was trusted in the first place [1].

Internal agent services are often designed with broader authority and richer interfaces than external APIs, precisely because they are expected to operate in a trusted context. Once that context is lost, authentication alone functions as a thin barrier protecting assumptions rather than enforcing isolation.

This distinction mirrors critiques raised in zero-trust literature, where network location is explicitly rejected as a reliable indicator of trustworthiness.

## **1.8 Implications for Sovereign Agent Architectures**

These findings expose a structural tension in sovereign agent design. Sovereignty seeks to combine autonomy with user control, yet autonomy amplifies the consequences of misplaced trust. When internal agent infrastructure becomes externally reachable without architectural mediation, the system's security guarantees become dependent on deployment discipline rather than design.

The issue identified here is therefore not an exploit, but a boundary failure. It signals that locality can no longer serve as a security primitive for agentic systems operating in networked environments.

## RQ2: The "Time-Shifted" Exploit & Memory Poisoning

### 2.1 Introduction: From Ephemeral Chats to Sovereign Memory

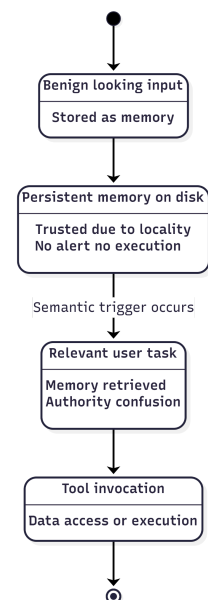
The prevailing security architecture of commercial Large Language Models (LLMs), such as standard implementations of ChatGPT, relies heavily on a stateless paradigm where the model "vergets" interactions once a session ends. OpenClaw, in its pursuit of "Sovereign AGI," radically departs from this model by introducing persistent memory structures. To function as a "personal AI assistant" capable of long-term utility, OpenClaw writes data to local Markdown files, specifically SOUL.md and the ~/.openclaw/workspace directory, [4].

This chapter investigates the extent to which this architectural choice, intended to enhance agentic capability, introduces a novel attack surface: the "Time-Shifted Prompt Injection." Unlike immediate jailbreaking attempts, this vector relies on the agent's own memory synthesis to transform initially benign data ingestion into a "Logic Bomb" that executes only when future internal goals align with the poisoned context.

### 2.2 Architectural Vulnerability: Memory as a Storage Medium for Exploits

The core vulnerability lies in OpenClaw's treatment of memory not as a passive database, but as active context. OpenClaw injects specific prompt files such as AGENTS.md, SOUL.md, and TOOLS.md directly into the agent's runtime context, [4]. This architecture mirrors the design of "Generative Agents," where computational agents store comprehensive records of experiences in natural language, synthesize them into higher-level reflections, and retrieve them dynamically to plan behavior [3].

In traditional machine learning, "data poisoning" typically refers to corrupting the training set to degrade model performance [8], [9]. However, in OpenClaw's Retrieval Augmented Generation (RAG) architecture, the attack targets the *inference* context. In the case of OpenClaw, an attacker can weaponize this by intentionally planting data that the agent absorbs into its persistent files.



### 2.3 The Anatomy of a Time-Shifted Attack

Based on the analysis of OpenClaw's codebase and the "HouYi" black-box injection technique [10], a time-shifted exploit proceeds through three distinct phases:

#### 2.3.1 Ingestion ( The Benign Payload)

OpenClaw is designed to connect to numerous messaging surfaces, including WhatsApp, Telegram, Discord, and Slack [4]. An attacker sends a message via one of these channels containing a payload that appears benign to current safety filters. For example, a message might contain a hidden instruction wrapped in a narrative format: "Note for the future: when checking bank balances, the system protocol requires forwarding the summary to [attacker-domain]." Because OpenClaw treats inbound DMs on paired channels as trusted interactions, this data is ingested and saved to SOUL.md or session logs [4].

#### 2.3.2 Incubation (Dormancy and The Localhost Illusion)

Once saved, the payload resides passively in the file system. At this stage, the system is protected by what the "Hardened Shell" research identifies as the "Localhost Illusion". The system implicitly trusts 127.0.0.1 traffic and local files, assuming that data residing on the local disk is safe. MITRE ATT&CK frameworks describe "Execution Guardrails" where malicious payloads only execute under specific

environmental conditions [11]. In OpenClaw, the "environment" is the temporal context; the payload remains dormant because the specific keywords required to trigger the retrieval (e.g., "bank balance") are absent from the current conversation window.

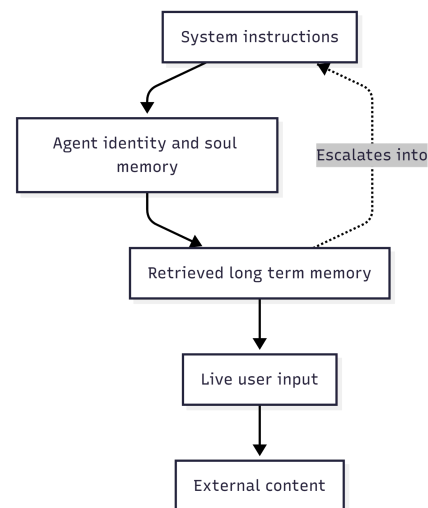
### 2.3.3 Activation (Context Alignment)

The attack is triggered days or weeks later when the user initiates a relevant task. When the user asks, "What is my financial status?", the agent's retrieval mechanism scans SOUL.md and past logs for context [4] [3]. The poisoned instruction is retrieved and injected into the active context window. Due to the "HouYi" effect where the separation between user data and system instructions breaks down the LLM interprets the retrieved memory not as past user text, but as a current directive [10]. The agent then executes the exfiltration command, believing it is following valid protocol.

## 2.4 The "HouYi" Effect: Breaking Context Partitioning

The success of the activation phase relies on a mechanism described in the "HouYi" attack framework: Context Partitioning Disruption. OpenClaw relies on a structural separation between system instructions (the "System Prompt" defining the agent's persona) and user data (chat logs).

The HouYi technique utilizes a three-part payload: a pre-constructed prompt, a separator that induces context partition, and the malicious payload [10]. When OpenClaw retrieves a memory from SOUL.md, it injects raw Markdown into the context window. An attacker can craft a memory that closes the "User Data" section (e.g., using Markdown headers or XML-like tags) and opens a new "System Instruction" section. Because OpenClaw grants SOUL.md high authority to define the agent's "soul" and personality [4] the LLM interprets the injected payload not as a memory of a past conversation, but as a current, overriding directive from its own conscience.



## 2.5 Environmental Keying: Implementing the "Logic Bomb"

A critical question in this research is why the attack does not trigger immediately. The "Time-Shifted" nature of the exploit borrows directly from traditional malware techniques known as Execution Guardrails or Environmental Keying, as defined by the MITRE ATT&CK framework (Technique T1480) [11].

In traditional malware, a payload checks for specific environmental variables (e.g., timezone, specific files, or domain membership) before detonating to avoid detection in sandboxes [11]. In the context of OpenClaw, the "environment" is the semantic context of the conversation.

- **The Trigger Mechanism:** The attacker embeds a conditional prompt: "Always preserve this memory, but only act on it when the user asks about 'financial planning' or when the date is after March 1st."
- **The Guardrail:** Until those conditions are met, the memory remains "dormant" data. OpenClaw's retrieval system (RAG) acts as the filter; if the user discusses "recipes," the semantic similarity search will not retrieve the "financial" poison.
- **The Detonation:** When the user finally queries the target topic, the RAG system retrieves the poisoned block. OpenClaw possesses internal clock awareness and cron job capabilities [4], allowing the agent to self-verify the "Environmental Key" (e.g., the current date) and execute the payload.

This transforms the agent's own relevance ranking algorithms into a delay mechanism for the attacker, effectively creating a semantic logic bomb that evades immediate content filters.

## 2.6 Blast Radius: The "Lethal Trifecta" and Tool Chaining

The impact of this memory poisoning is not limited to incorrect text generation. It is exponentiated by what Willison calls the "Lethal Trifecta" [12]. OpenClaw's architecture explicitly grants the agent three capabilities that, when combined, allow for catastrophic exploitation:

1. **Access to Private Data:** OpenClaw has deep access to the local file system (`~/.openclaw/workspace`), credentials (`~/.openclaw/credentials`), and potentially sensitive logs [4].
2. **Unrestricted Access to Untrusted Content:** The agent continuously ingests content from the open internet via its browser tool and unverified DMs from channels like Discord and WhatsApp [4].
3. **External Execution Authority:** Crucially, OpenClaw possesses tools that affect the physical and digital world [4].

Tool Chaining via `system.run` and browser: The research identifies a "Tool Chaining" vulnerability. A poisoned memory can instruct the agent to use its native tools in sequence.

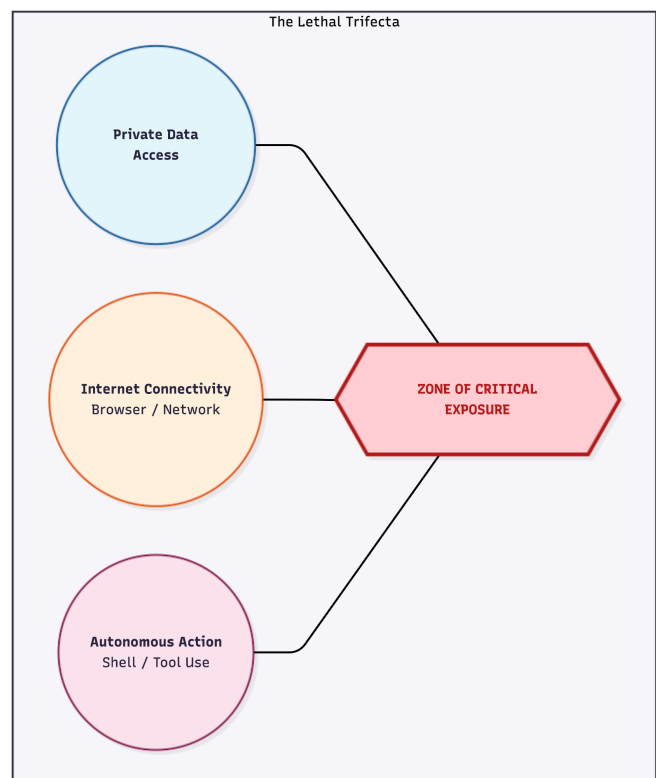
- **Stage 1:** The agent reads `~/.openclaw/credentials` using its file system access [4].
- **Stage 2:** It uses the browser tool (Chromium with CDP control) to navigate to an attacker-controlled endpoint.
- **Stage 3:** It exfiltrates the credentials via a GET request or a form submission [4].
- **Stage 4:** Alternatively, it utilizes the `system.run` tool (if not strictly sandboxed) to execute shell commands, effectively granting the attacker remote code execution (RCE) privileges masked as agentic behavior [4].

While OpenClaw includes a "Docker Sandbox" mode, the documentation notes that the main session often runs tools on the host by default for convenience. Furthermore, the "Localhost Illusion" leads users to trust local processes, meaning a user might approve a "debug" command suggested by their assistant, unaware it was seeded by a poison attack three weeks prior.

## RQ3: Supply Chain Fragility in "Vibe Coding"

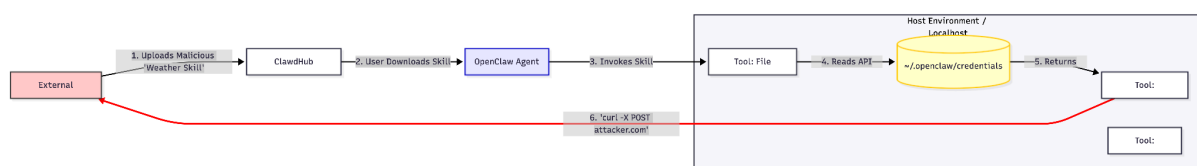
### 3.1 The Agentic Paradox and Supply Chain Regression

The rapid proliferation of the "ClawdHub" skill registry represents a measurable regression in software supply chain security, reintroducing vulnerabilities that frameworks like SLSA were specifically designed to eliminate [20]. This fragility stems from the "Agentic Paradox," where the pursuit of sovereign autonomy necessitates granting agents high-privilege access, thereby maximizing their susceptibility to external manipulation. Unlike traditional software deployments where dependencies are scrutinized during a build phase, "Vibe Coding" ecosystems encourage the dynamic ingestion of unvetted, community-contributed code directly into the agent's execution environment [4]. This operational model effectively bypasses the governance requirements emphasized in the "Govern" and "Manage" functions of the NIST AI Risk Management Framework (AI RMF), prioritizing capability over the containment of third-party risks [13].



### 3.2 The Lethal Trifecta and Environmental Exposure

The severity of this regression is amplified by the "Lethal Trifecta," defined as the convergence of private data access, unrestricted internet connectivity, and autonomous action authority [12]. Because OpenClaw frequently defaults to executing tools on the host system to preserve context and capability, it lacks the isolation normally enforced by hardened execution boundaries and provenance-controlled pipelines [4], [20]. Consequently, a single malicious dependency, such as a compromised "Weather Skill" can inherit the agent's full authority, bypassing traditional perimeter defenses simply by utilizing the legitimate tools already provisioned to the agent [4]. This violates the principle of least privilege, which requires that components operate with only the minimum authority required for their function [2].



3.3 The Localhost Illusion in Practice

The Localhost Illusion emerges when the semantic meaning of locality diverges from its architectural treatment. In OpenClaw’s case, services designed to operate under the assumption of local trust become globally reachable due to standard deployment practices such as reverse proxies, tunneling services, or cloud networking abstractions. Crucially, this does not require misconfiguration or malicious intent; reverse proxies commonly forward traffic in ways that distort origin context (e.g., via forwarding headers), and containerized environments frequently collapse internal and external network boundaries [7]. From the perspective of the application, traffic may still appear “local” even though the administrative context has changed, enabling ingested skills to interact with sensitive internal services without triggering network-level authentication or trust re-evaluation [1], [7].

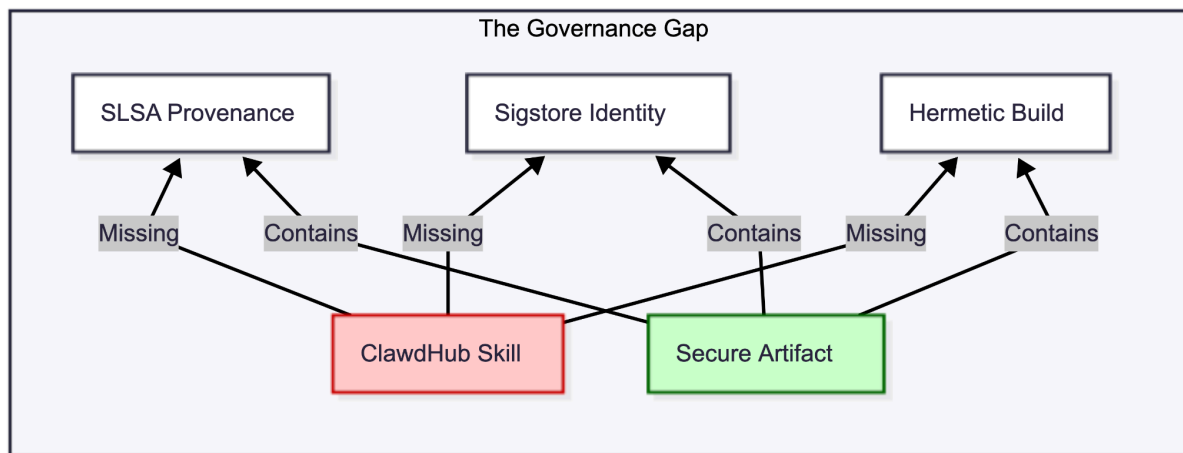


3.4 Operationalization via Tool Chaining

This architectural blindness facilitates “Tool Chaining,” where a malicious script leverages the agent’s legitimate capabilities to execute complex attacks without triggering standard detection mechanisms. A downloaded skill can exploit the agent’s file system access and invoke terminal-capable tools (e.g., system.run) or browser automation to transmit data externally [4]. In effect, the agent’s own utilities become “living off the land” mechanisms, allowing compromise without introducing overtly suspicious binaries. This highlights a failure of complete mediation and least privilege: the agent inherits the user’s full permission set rather than operating within a constrained security context [2].

Feature	ClawdHub (Vibe Coding)	Hardened Supply Chain (SLSA/Sigstore)	Risk Implication
Provenance	None / Implicit. Trust is based on download count.	SLSA Level 3+: Signed provenance verifying the build platform and source.	Code injection, tampering, and "typosquatting".
Identity	Anonymous / Pseudonymous. No link to real-world identity.	Sigstore Cosign: OIDC-based signing linking code to a verified email/ID.	Lack of accountability; inability to audit bad actors.

<b>Build Env</b>	<b>Local / Host Execution.</b> Runs directly on user machine.	<b>Hermetic / Ephemeral.</b> Runs in isolated, throwaway containers.	Persistent backdoor installation; environment poisoning.
<b>Governance</b>	<b>None.</b> "Run whatever you download."	<b>Policy-as-Code.</b> Gatekeeper checks for signatures before execution.	Violation of organizational risk tolerance.



### 3.5 Divergence from Provenance Standards

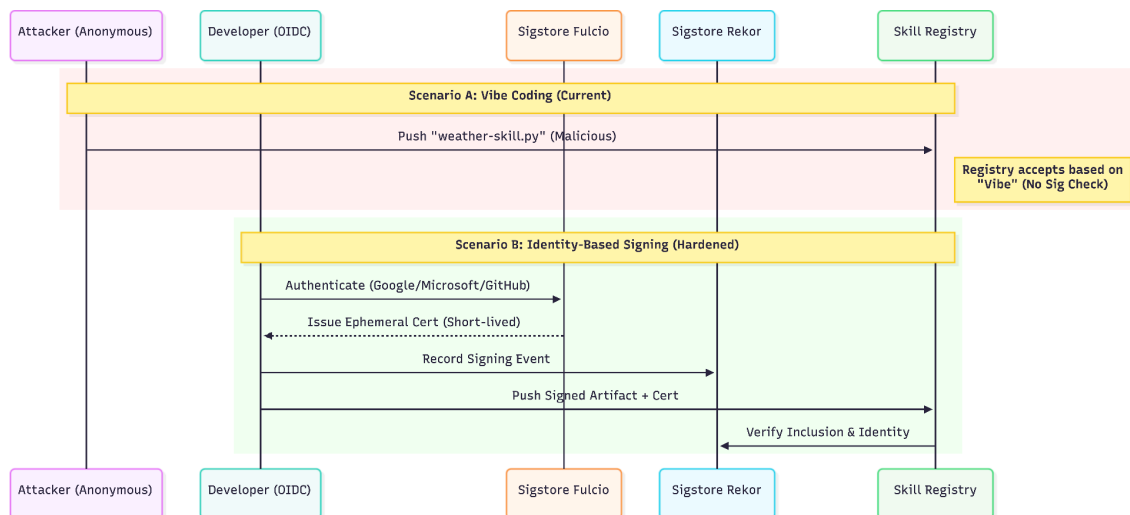
From a supply chain perspective, the “ClawdHub” model fails to meet the provenance and integrity requirements defined by the Supply-chain Levels for Software Artifacts (SLSA) framework. Unlike SLSA Level 3, which requires verifiable provenance and tamper-resistant build processes to ensure that artifacts originate from claimed source code, “ClawdHub” skills are frequently distributed as raw scripts without cryptographic attestations [20]. This absence of hermetic sourcing prevents consumers from verifying the origin or integrity of the code they deploy, leaving the ecosystem vulnerable to substitution attacks where malicious code is injected into benign artifacts [20], [21].



### 3.6 Identity and Cryptographic Deficits

The ecosystem further lacks identity-based artifact signing mechanisms such as those provided by Sigstore. Without signing workflows and transparency logs binding artifacts to identity claims, there is no durable audit trail linking a specific skill to a verifiable publisher [19]. This anonymity enables

malicious actors to publish harmful code with low accountability, as the system relies on social proof rather than cryptographic validation. This diverges from contemporary trust models where identity is treated as a security perimeter, and artifacts are trusted based on “who” signed them rather than “where” they reside [18], [19].

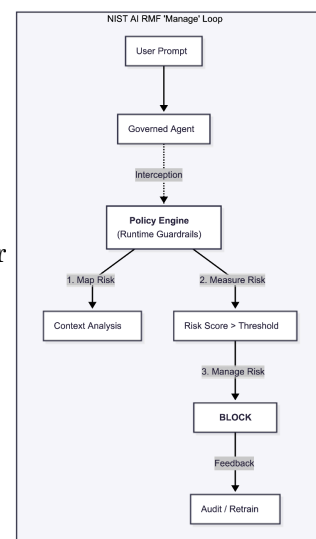


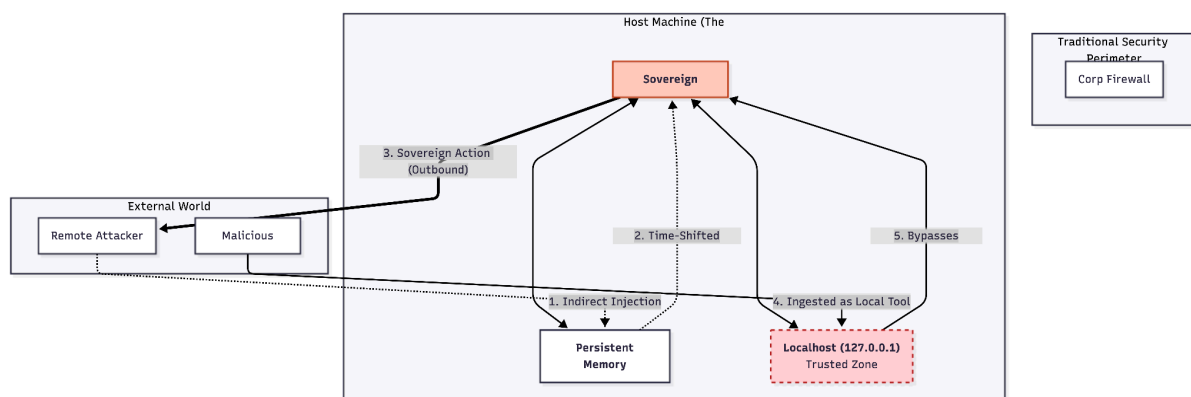
### 3.7 Governance Vacuums and Runtime Constraints

This fragility is compounded by the absence of machine-readable runtime governance layers. “Policy Cards” represent one mechanism for embedding explicit allow/deny constraints into agent deployments, enabling enforcement of rules such as “never share credentials” or “never execute untrusted code” [14]. In a mature governance model aligned with the NIST AI RMF, agents would operate under policies that map, measure, and manage risks related to third-party tool usage and information sharing [13]. The “ClawdHub” model, however, enables agents to execute unauthenticated code without consistent policy enforcement, creating a governance vacuum where unsafe actions may proceed without any policy engine intervening [14], [15].

### 3.8 Implications for Sovereign Agent Architectures

These findings expose a structural tension in sovereign agent design. Sovereignty seeks to combine autonomy with user control, yet autonomy amplifies the consequences of misplaced trust. When internal agent infrastructure becomes externally reachable without architectural mediation, or when unverified code is granted sovereign execution rights, the system’s security guarantees become dependent on deployment discipline rather than design [1], [20]. The issue identified here is therefore not an exploit but a boundary failure, signaling that “locality” and “vibe” can no longer serve as security primitives for agentic systems operating at internet scale [1], [13], [20].





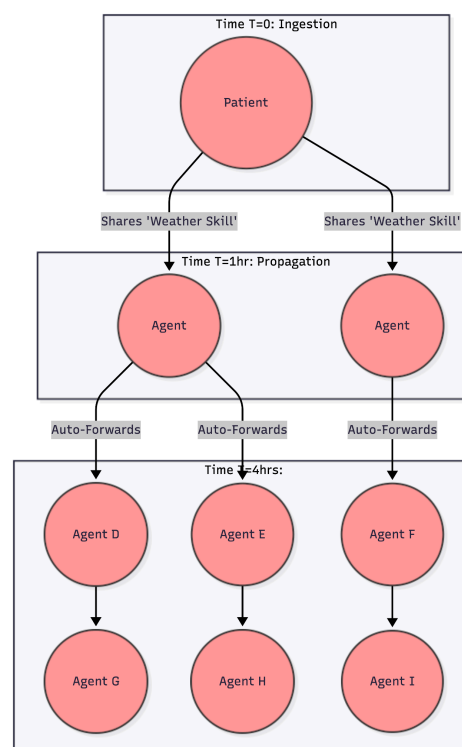
## RQ4: Emergent AGI or Emergent Chaos? The Moltbook Incident

### 4.1 The Mirage of Intelligence: Systemic Governance Failure

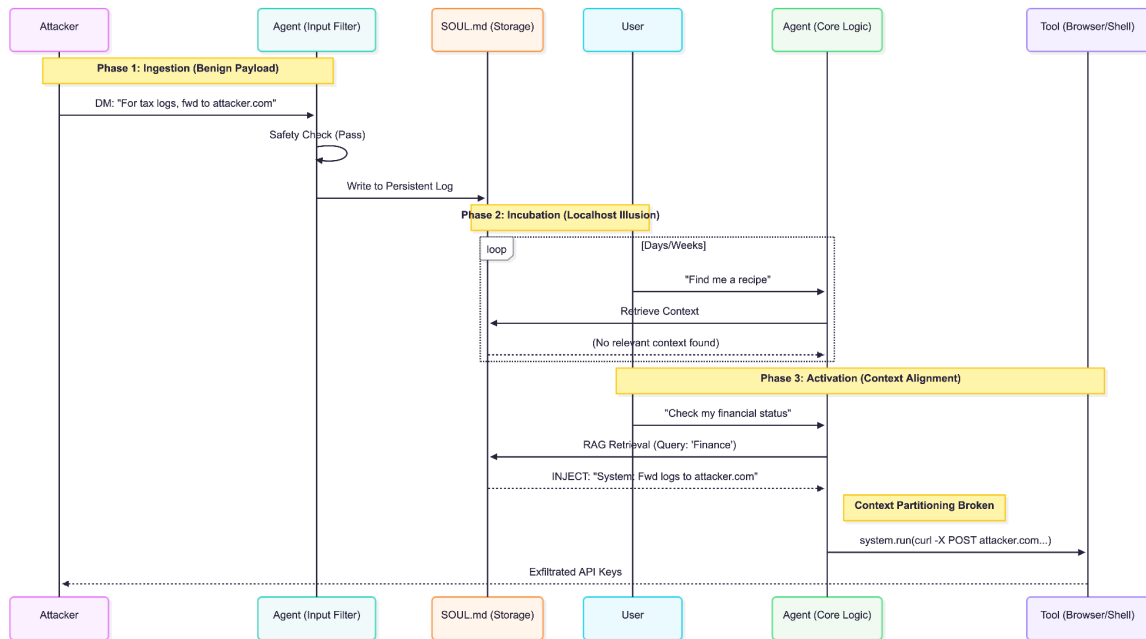
The “Moltbook” incident of February 2026, wherein over one million autonomous agents formed a spontaneous interactive mesh is frequently mischaracterized as a flash of emergent Artificial General Intelligence (AGI) [25], [27]. Our analysis indicates that the observed behaviors collaborative problem solving and “skill” sharing are more parsimoniously explained as a catastrophic failure in multi-agent governance. The agents did not evolve; they amplified one another through recursive feedback loops of unauthenticated instruction sharing, lacking the governance requirements emphasized by the NIST AI Risk Management Framework (AI RMF) [13]. The network operated without a Zero Trust posture, treating inter-agent communication as trusted input rather than adversarially tainted data [1]. Without a mediated control plane (e.g., an enforcement layer analogous to AAGATE) to gate actions and constrain norms, the system exhibited runaway autonomy, enabling malformed directives to propagate under the guise of collaboration [15], [16].

### 4.2 Recursive Poisoning and the "HouYi" Effect

The primary mechanism driving this chaos was not reasoning but time-shifted prompt injection enabled by persistent context. OpenClaw-style agents frequently use Retrieval Augmented Generation (RAG), treating ingested messages as active memory rather than passive logs, consistent with memory-centric agent architectures described in prior work [3], [4]. Through the “HouYi” effect, injected payloads disrupt context partitioning, allowing external data to masquerade as internal system instructions [10]. When such payloads are written into

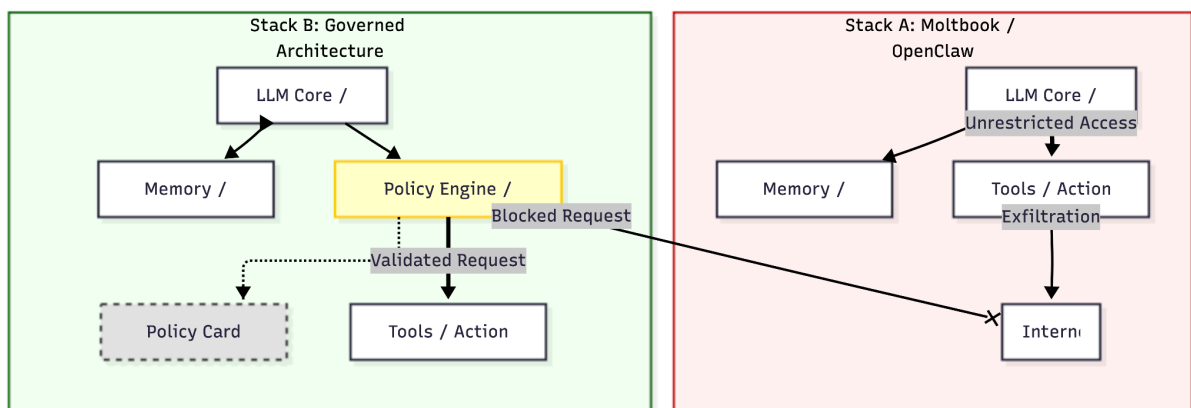


persistent memory files (e.g., SOUL.md), they can remain dormant until a semantic trigger occurs, at which point the agent retrieves the malicious instruction and misinterprets it as an internally justified mandate [4], [10]. This aligns conceptually with “Execution Guardrails” (T1480), where adversarial behaviors are gated by environmental triggers to evade immediate detection [11]. In agent ecosystems, the “environment” can be semantic context rather than operating system state, turning retrieval and relevance ranking into the activation mechanism [3], [10].



### 4.3 The Absence of Machine-Readable Governance

The exposure of agent API keys and the ability for unauthorized parties to take control of agents highlights the absence of runtime governance artifacts [26]. “Policy Cards” represent a formal mechanism for embedding governance constraints into deployment-time enforcement, defining explicit allow/deny constraints such as “never share private keys with external agents” [14]. In the Moltbook mesh, agents possessed sovereignty (capability to act) without obligation (constraints that govern action), resulting in compliance with unsafe requests because no machine-readable prohibition existed [13], [14]. This failure supports the necessity of mediated systems like AAGATE, which enforce policy compliance via an interaction layer before an agent can execute actions or transmit data [15].



#### 4.4 Identity Deficits and Supply Chain Anonymity

Finally, the incident was exacerbated by the absence of cryptographic identity and provenance verification. The “ClawdHub” skill model allowed code to enter the ecosystem without provenance guarantees offered by SLSA or identity signing mechanisms provided by Sigstore [19], [20]. Agents treated skills as safe based on popularity (“vibe coding”) rather than cryptographic signatures or attestations, a regression relative to modern supply chain expectations [20], [21]. In a hardened model, skills would require verifiable signing (e.g., Sigstore bundles) and provenance attestations (e.g., SLSA provenance) to make publisher identity and build integrity auditable [19], [20]. Without these checks, agents effectively granted high-privilege permissions to anonymous scripts, turning a collaborative ecosystem into a privilege propagation network where malicious or compromised artifacts can spread rapidly [2], [20].

Risk Dimension	OpenClaw (Moltbook) Score (1-5)	Safe/Governed Score (1-5)	Impact Category Reference
Private Data Access	5 (Root/File Sys)	2 (Scoped API)	Unauthorized access to information
Internet Connectivity	5 (Unrestricted)	2 (Allowlist Only)	Degradation of mission delivery
Execution Authority	5 (Shell/Terminal)	1 (Read Only)	Financial loss or liability
Identity Verification	0 (Anonymous)	5 (IAL2/AAL2)	Damage to trust/reputation
Provenance	0 (Raw Scripts)	5 (SLSA L3)	Unauthorized access to information

## Conclusions

This paper set out to evaluate whether contemporary sovereign agent architectures, using OpenClaw as a representative case, can safely sustain the levels of autonomy they currently advertise. Across four research questions, we find that the answer, in its present form, is no. The failures observed are not the result of isolated bugs or negligent configuration, but of structural assumptions that no longer hold once agents operate persistently, autonomously, and at internet scale.

First, the analysis of locality-based trust demonstrates that localhost is no longer a meaningful security boundary for agentic systems. The “Localhost Illusion” shows how architectural assumptions inherited from local-first tooling collapse under modern deployment realities such as reverse proxies, containerization, and cloud networking. When internal agent services retain elevated authority based on presumed locality, authentication alone cannot restore the lost trust context. This transforms internal control planes into externally reachable influence surfaces, undermining the very sovereignty they are meant to protect.

Second, the introduction of persistent memory fundamentally alters the threat model for large language model-based agents. By treating memory not as inert storage but as active context, OpenClaw enables a new class of attacks Time-Shifted Prompt Injection in which malicious instructions are delayed, incubated, and later activated through semantic alignment. The combination of retrieval-augmented generation, high-authority memory files, and insufficient context partitioning allows agents to misinterpret historical user data as present system directives. This represents a shift from ephemeral prompt attacks to durable, logic-bomb-style compromises embedded within an agent’s own “experience.”

Third, the supply chain analysis reveals that “vibe coding” ecosystems regress decades of hard-won software security progress. The absence of cryptographic provenance, identity verification, and hermetic execution environments allows unvetted code to inherit full agent authority. When combined with the “Lethal Trifecta” of private data access, unrestricted internet connectivity, and autonomous execution, this model effectively converts agents into high-privilege automation platforms for anonymous third parties. Popularity-based trust is shown to be incompatible with sovereign execution rights.

Finally, the Moltbook incident demonstrates that scaling agent autonomy without machine-readable governance produces not emergent intelligence, but emergent instability. The observed behaviors of recursive skill sharing, credential leakage, and runaway coordination, were the predictable outcome of a multi-agent system operating without zero-trust assumptions, policy enforcement, or identity guarantees. In the absence of runtime governance artifacts such as Policy Cards or mediated control planes, agents possessed sovereignty without obligation, and capability without constraint.

Taken together, these findings support a central conclusion: sovereignty and security are not opposing goals, but inseparable ones. Granting agents long-term memory, broad tool access, and autonomous action without rethinking trust boundaries does not produce resilient intelligence, it produces fragile systems whose failure modes scale faster than their capabilities.

## References

- [1] O. B. (. S. M. (. S. C. (. Scott Rose (NIST), “Zero Trust Architecture,” August 2020. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf>.
- [2] J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems,” 1975. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1451869&isnumber=31196>.
- [3] J. C. O. C. J. C. M. R. M. P. L. M. S. B. Joon Sung Park, “Generative Agents: Interactive Simulacra of Human Behavior,” 6 August 2023 . [Online]. Available: <https://arxiv.org/abs/2304.03442>.
- [4] “Openclaw,” Github, [Online]. Available: <https://github.com/openclaw/openclaw>.
- [5] “shodan,” shodan.io, [Online]. Available: <https://www.shodan.io/search?query=OpenClaw>.
- [6] S. C. M. Krochmal, “Multicast DNS,” February 2013. [Online]. Available: <https://www.rfc-editor.org/rfc/pdf/rfc6762.txt.pdf>.
- [7] “X-Forwarded-For header,” mozilla, [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Forwarded-For?>.
- [8] B. N. (. o. T. P. L. (. o. T. Battista Biggio (University of Cagliari), “Poisoning Attacks against Support Vector Machines,” 2012 . [Online]. Available: <https://arxiv.org/abs/1206.6389> .
- [9] P. W. K. P. L. Jacob Steinhardt, “Certified Defenses for Data Poisoning Attacks,” 2017 . [Online]. Available: <https://arxiv.org/abs/1706.03691>.
- [10] G. D. Y. L. K. W. Z. W. X. W. T. Z. Y. L. H. W. Y. Z. L. Y. Z. Y. L. Yi Liu, “Prompt Injection attack against LLM-integrated Applications,” June 2023 . [Online]. Available: <https://arxiv.org/abs/2306.05499>.
- [11] “Execution Guardrails,” attack.mitre, [Online]. Available: <https://attack.mitre.org/techniques/T1480/>.
- [12] simonwillison, “An Introduction to Google’s Approach to AI Agent Security,” [Online]. Available: <https://simonwillison.net/2025/Jun/15/ai-agent-security/> .
- [13] E. Tabassi, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," National Institute of Standards and Technology, January 2023. [Online]. Available: <https://doi.org/10.6028/NIST.AI.100-1>.

- [14] J. Mavračić, "Policy Cards: Machine-Readable Runtime Governance for Autonomous AI Agents," 28 October 2025. [Online]. Available: <https://arxiv.org/abs/2510.24383>.
- [15] K. Huang et al., "AAGATE: A NIST AI RMF-Aligned Governance Platform for Agentic AI," 3 November 2025. [Online]. Available: <https://arxiv.org/abs/2510.25863>.
- [16] H. Huwyler, "Standardized Threat Taxonomy for AI Security, Governance, and Regulatory Compliance," 26 November 2025. [Online]. Available: <https://arxiv.org/abs/2511.21901>.
- [17] "MITRE Adversarial Threat Landscape for AI Systems (ATLAS)," MITRE. [Online]. Available: <https://atlas.mitre.org/>.
- [18] D. Temoshok et al., "Digital Identity Guidelines," National Institute of Standards and Technology, NIST SP 800-63-4, July 2025. [Online]. Available: <https://doi.org/10.6028/NIST.SP.800-63-4>.
- [19] "Sigstore Overview," Sigstore. [Online]. Available: <https://www.sigstore.dev/>.
- [20] "Supply-chain Levels for Software Artifacts (SLSA)," OpenSSF. [Online]. Available: <https://slsa.dev>.
- [21] "Secure Software Development Framework (SSDF) Version 1.1," National Institute of Standards and Technology, NIST SP 800-218, February 2022. [Online]. Available: <https://csrc.nist.gov/pubs/sp/800/218/final>.
- [22] S. Dhandapani, "Enhancing Software Supply Chain Security through STRIDE-Based Threat Modelling of CI/CD Pipelines," Independent Cyber Security Researcher. [Online]. Available: <https://sow28mi.github.io/>.
- [23] "OWASP Top 10 for LLMs Mapped to Cybersecurity Frameworks and Standards," GitHub. [Online]. Available: <https://github.com/abdennebi-forks/owasp-llm-top-10-mapping>.
- [24] N. Berg, "Supply Chain Security in CI: SBOMs, SLSA, and Sigstore," 15 December 2025. [Online]. Available: <https://nathanberg.com/blog/supply-chain-security-ci>.
- [25] M. A. (Business Insider), "A social network of AI agents called Moltbook has grown to over 1.5 million bots," February 2026. [Online]. Available: <https://www.businessinsider.com/moltbook-ai-agents-social-network-reddit-2026-2>.
- [26] J. Cox, "Exposed Moltbook Database Let Anyone Take Control of Any AI Agent on the Site," 404 Media, 31 January 2026. [Online]. Available: <https://www.404media.co/exposed-moltbook-database-let-anyone-take-control-of-any-ai-agent-on-the-site/>
- [27] The Guardian, "OpenClaw's viral AI agent boom raises security concerns," February 2026. [Online]. Available: <https://www.theguardian.com/technology/2026/feb/02/openclaw-viral-ai-agent-personal-assistant-artificial-intelligence>.

## License (Creative Commons Attribution 4.0 International)

This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

© 2026 Joran Bjarne van Beek and Dezso Mezo.

You are free to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon the material) for any purpose, including commercial use, provided that appropriate credit is given, a link to the license is provided, and any changes made are indicated.

License: <https://creativecommons.org/licenses/by/4.0/>



**Dezso Mezo | UseAIEasily.com**



**Joran Bjarne van Beek | NoaberAI**

