OpenClaw Ecosystem Security Report

Jiahao Chen[1], **Bingduo Liao**[2], **Shixuan He**[2], **Xinfeng Li**[2], **Shouling Ji**[1]

[1]Zhejiang University, [2]Nanyang Technological University

OpenClaw has rapidly evolved into a public general-purpose agent ecosystem. As of March 26, 2026, the core repository had reached 337K GitHub stars and the associated ClawHub registry exposed more than 37.2K published skills. Yet current security understanding still concentrates on single layers such as malicious skills, runtime attacks, or social traces, rather than the coupled chain through which risky artifacts are published, updated, promoted, installed, exposed, and only later contained. This report therefore documents OpenClaw as an ecosystem security problem and grounds its claims in public, reproducible, and provenance-preserving artifacts. Our current evidence bundle combines versioned registry data, normalized incident and vulnerability records, GitHub security and release activity, and confidence-aware community traces. Unless explicitly noted otherwise, the evidence-bearing collections used in this report are frozen through March 18, 2026. Within that bundle, the analyzed snapshot already covers 21,420 distinct skills, 26,984 observed versions, 83 public incident reports, and 91 CVE disclosures.

The current evidence supports four report-level observations. First, the threat landscape is already ecosystem-scale rather than artifact-scale: registry activity, incidents, vulnerabilities, and response artifacts all appear in the same public security chain. Second, risk is lifecycle-structured: most eventually risky skills are already risky at first release, while the benign-then-risky subset is small, transitions early, and shows a median poisoning delay of about 57 minutes. Third, propagation beyond the registry is observable but uneven: among 1,884 promotion events, only 46 qualify as high-confidence artifact links, so propagation should be treated as a confidence-aware structural signal rather than as causal proof. Fourth, containment is visible as a downstream stage, which makes layered control-point reasoning possible across publish-time, update-time, propagation-time, and response-time defenses. Taken together, these results support viewing OpenClaw as a coupled security ecosystem, while also making clear that stronger prevalence and causal claims still require a more complete longitudinal corpus.

Our data and website are available as well.

Date: April 2026
Contact: Xinfeng Li: lxfmakeit@gmail.com

1 Introduction

OpenClaw has rapidly evolved from a local agentic assistant into a representative *General-purpose agentic assistant (GAA)* ecosystem. As of March 26, 2026, the OpenClaw core repository had reached 337K GitHub stars and the associated ClawHub registry exposed more than 37.2K published skills OpenClaw (2026e,c). This scale no longer describes a standalone assistant binary. It describes an ecosystem that spans the core runtime OpenClaw (2026e), versioned skills OpenClaw (2026c), tool and plugin integrations OpenClaw (2026e), and public community spaces such as Moltbook Moltbook (2026). ClawHub OpenClaw (2026a,d) exposes skills as versioned and downloadable artifacts with ranking and installation-oriented telemetry, while Moltbook and adjacent channels expose traces of recommendation, imitation, and downstream diffusion Moltbook (2026); TrustAIRLab (2026). Together, these layers reflect a broader shift in GAAs toward coupling executable capabilities, third-party extensions, continuous updates, and public discovery mechanisms in one user-facing stack Oltrogge et al. (2018).

However, the same properties that make such ecosystems useful also create a new security reality. Public

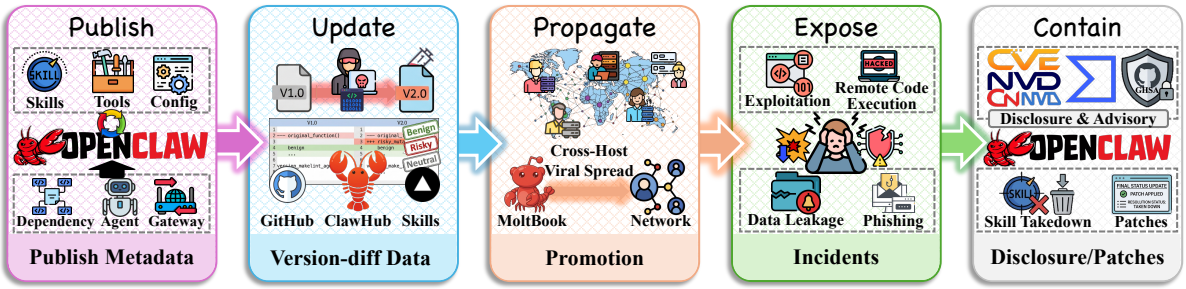


Figure 1 OpenClaw-centered ecosystem chain.

reports have already documented campaign-style malicious skill waves such as ClawHavoc Koi Security (2026), while exposure reporting has described more than 135,000 publicly exposed OpenClaw instances and Moltbook-related data leakage SecurityScorecard (2026); Wiz (2026). The current repository snapshot also contains normalized OpenClaw-related NVD records dated from February 1, 2026 through February 19, 2026, covering command injection, code injection, authentication or authorization failures, network-boundary flaws, and data exposure National Institute of Standards and Technology (2026). Taken together, these cases suggest that GAA insecurity is better understood as an ecosystem-level failure mode than as a collection of isolated bugs or isolated bad skills. Risk can be published, intensified through later updates, amplified through public propagation or deployment, and only then become visible through disclosure and containment artifacts Koi Security (2026); National Institute of Standards and Technology (2026); OpenClaw (2026f).

That lifecycle also exposes the core gap that motivates this paper. Recent security studies have established important pieces of the OpenClaw landscape, but mostly one layer at a time. At the skill layer, Liu et al. Liu et al. (2026) measures vulnerabilities in agent skills at scale, *SkillJect* Jia et al. (2026) shows that skill-based prompt injection can be systematically weaponized, and *SkillFortify* Bhardwaj (2026) frames agent skills as a supply-chain security problem. At the OpenClaw runtime layer, prior studies benchmark adversarial scenarios and report only a 17% native defense rate across 47 attacks, formalize personalized-agent attack surfaces, or decompose OpenClaw threats into lifecycle stages such as supply-chain contamination, memory poisoning, and risky execution Shan et al. (2026); Wang et al. (2026); Deng et al. (2026); Dong et al. (2026). At the social layer, Moltbook studies show that risky instruction sharing, participation inequality, and large-scale community dynamics are measurable in OpenClaw's public community ecosystem Manik and Wang (2026); Chen et al. (2026). These studies provide valuable baselines, but their scopes remain bounded to prevalence-oriented scanning, attack generation, runtime evaluation, or community characterization rather than the full ecosystem chain. As a result, we still lack a unified, evidence-bounded account of the ecosystem-wide threat landscape, version-aware risk evolution, cross-platform propagation, and defense leverage across the same OpenClaw security chain.

OpenClaw is a strong case for closing these gaps because the ecosystem exposes rich observability across the same chain that defines the problem. It exposes versioned skill artifacts and installation-oriented registry metadata in ClawHub OpenClaw (2026b,d), public incident reports and NVD disclosures Koi Security (2026); National Institute of Standards and Technology (2026), post-incident GitHub hardening activity OpenClaw (2026f), and public community traces relevant to Moltbook and adjacent channels Moltbook (2026); TrustAIRLab (2026). This makes it possible to study not only where risk appears, but also how it evolves, propagates, and becomes visible to defenders. More importantly, many GAAs share the same structural pressure points: low-friction extension distribution, trust that can be established before later risky updates, public recommendation or social-discovery channels, and delayed containment after compromise signals become public Oltrogge et al. (2018); Hou et al. (2025). We therefore use OpenClaw as both a concrete analysis case and an entry point for understanding the broader failure geometry of GAAs.

Guided by this case-to-class perspective, we organize the paper around four research questions. **RQ1**: What is the threat landscape of the OpenClaw ecosystem? **RQ2**: How does security risk evolve across the ecosystem lifecycle? **RQ3**: How does security risk propagate across the OpenClaw ecosystem? **RQ4**: What defense leverage points can most effectively reduce ecosystem risk? Together, these questions follow the same narrative backbone: *publish* → *update* → *propagate* → *expose* → *contain*. We answer these questions by treating

OpenClaw security as an ecosystem-chain analysis problem rather than a single-point vulnerability analysis. Figure 1 summarizes this conceptual model. Risk can be introduced at publication time, intensified through later updates, propagated through public promotion and cross-host circulation, realized through installation or exposed deployments, and only later countered through takedowns, disclosures, or security hardening. This framing is directly motivated by current public evidence: normalized incident reports already support campaign-style abuse as a meaningful unit of analysis Koi Security (2026), the current NVD records show that platform-level failures span multiple bug classes and dates National Institute of Standards and Technology (2026), and the March 9, 2026 release line documents downstream hardening for SSRF, authorization, command-execution safety, and skill-download controls OpenClaw (2026f).

Specifically, we build an automated, provenance-first evidence pipeline that ingests and links public artifacts across the OpenClaw ecosystem. The pipeline aligns external skill-risk assessments with versioned skill artifacts, incident and vulnerability disclosures, GitHub security and release activity, and community traces; reconstructs version-aware evidence paths; clusters campaign signals across publishers, timing, and artifacts; links propagation events back to risky artifacts; and supports retrospective analysis of where publish-time, update-time, propagation-time, and containment-time controls could have intervened OpenClaw (2026b,d); Socket (2026); Snyk (2026). This design preserves provenance at every stage and allows the report to scale as the longitudinal corpus becomes more complete, without converting partial collections into unsupported prevalence or causal-impact claims. Unless explicitly noted otherwise, the public evidence used in this report is collected and frozen through March 18, 2026.

In summary, we make the following contributions:

1. We formulate OpenClaw security as an ecosystem-chain analysis problem and use it to motivate a broader security lens for GAAs. This framing treats publication, update, propagation, exposure, and containment as coupled stages of one observable failure process, and it organizes the study around four research questions on threat landscape, lifecycle evolution, ecosystem propagation, and defense leverage points.
2. We design an automated, provenance-first evidence pipeline for answering these questions with public evidence. The pipeline links versioned skill artifacts, incident reports, NVD disclosures, GitHub release and security activity, and community traces to support version-graph reconstruction, campaign clustering, cross-platform linkage, and retrospective intervention analysis while keeping evidence and inference explicitly separated.
3. We provide an evidence-bounded security report on OpenClaw ecosystem risk. The current results show that risk is structurally concentrated rather than random, that later poisoning exists but is rare and fast, that propagation is observable but dominated by a small high-confidence subset, and that no single control point is sufficient for ecosystem defense.

2 Background and Related Work

OpenClaw Architecture and Ecosystem. The ecosystem-chain view introduced in Section 1 follows directly from how OpenClaw is built and used in practice. OpenClaw is not merely a local assistant binary, but a layered ecosystem in which an agent runtime, extensibility mechanisms, public distribution channels, community-mediated propagation, and real deployment surfaces interact in one operational stack OpenClaw (2026e,c); Moltbook (2026). The runtime and capability layers define what the system can perceive and execute, the registry layer defines how artifacts are published and updated, the social layer defines how those artifacts gain visibility and legitimacy, and the exposure and response layers define how risk becomes operationally visible and eventually contained. This is why the paper studies OpenClaw at the ecosystem level rather than as a single software component: *risk can move across layers that are technically distinct and socially coupled.*

ClawHub as a Supply Chain. Within this layered ecosystem, ClawHub functions as the main supply-chain substrate for skills rather than as a simple extension catalog OpenClaw (2026b,d). It is where skills are packaged, published, versioned, discovered, and reused, and for each skill it exposes the metadata, publisher identity, popularity signals, and version history needed for longitudinal analysis. By preserving version-level evidence, ClawHub makes it possible to distinguish publication risk from lifecycle risk and to ask whether a skill was risky from first release or became risky only after trust and visibility had already been established. In

our analysis design, this versioned registry evidence is not used as a standalone risk oracle; it is the temporal backbone against which externally observed risk assessments and incident signals are aligned.

Community Propagation Layer. OpenClaw's ecosystem also includes a public propagation layer, with Moltbook as the clearest native example since it exposes agent-authored posts, interactions, and recommendation traces Moltbook (2026); TrustAIRLab (2026). Additional channels such as Reddit, Hacker News, and Telegram extend this layer beyond the native platform. From an ecosystem-analysis perspective, these channels matter because they expose not only discussion, but also signals of circulation such as skill references, install commands, repeated templates, and duplicated posts that can be linked back to concrete artifacts. This is why the propagation layer should be treated as part of the security problem rather than as background noise, while still requiring explicit linkage and confidence-aware analysis instead of broad thematic similarity alone.

Known Security Incidents in the OpenClaw Ecosystem. OpenClaw already exhibits incident-grounded evidence of ecosystem failure. Public reporting has documented malicious-skill campaigns such as ClawHavoc, suspicious skill clusters, insecure deployments, exposed endpoints, and leaked credentials Koi Security (2026); SecurityScorecard (2026); Wiz (2026), while the current repository also contains normalized OpenClaw-related NVD records spanning multiple vulnerability classes, including command injection, code injection, authentication or authorization failures, network-boundary flaws, and data exposure National Institute of Standards and Technology (2026). These incidents matter because they show that the security problem is not reducible to a single attack primitive or artifact family. Just as importantly, the response side is also publicly observable: GitHub advisories, issues, commits, and security-oriented release notes show that maintainers continued to harden the platform after public signal appeared, giving the paper an observable containment stage rather than an inferred one OpenClaw (2026f).

Current OpenClaw Security Literature. Existing OpenClaw-related security research already shows that each major ecosystem layer is worth studying. At the skill layer, *Agent Skills in the Wild*, *SkillJect*, and *SkillFortify* demonstrate that skill ecosystems can contain risky instructions, prompt-injection surfaces, and broader supply-chain abuse patterns Liu et al. (2026); Jia et al. (2026); Bhardwaj (2026). At the runtime layer, prior studies benchmark adversarial scenarios, quantify weak native defense performance, formalize personalized-agent attack surfaces, or decompose OpenClaw risk into lifecycle stages such as supply-chain contamination, memory poisoning, and risky execution Shan et al. (2026); Wang et al. (2026); Deng et al. (2026); Dong et al. (2026). At the community layer, Moltbook studies show that risky instruction sharing, participation inequality, and large-scale public dynamics are measurable Manik and Wang (2026); Chen et al. (2026). These studies provide strong baselines, but they remain bounded in scope: skill-centric work usually stops before version evolution and propagation, runtime-centric work usually stops before registry and community linkage, and social-layer work usually stops before version-level risk and containment.

Positioning and Differentiation. This paper's positioning follows directly from the gaps above. Our contribution is to connect the pieces into one coupled analysis frame covering threat landscape, lifecycle evolution, ecosystem propagation, and defense leverage. This is why OpenClaw is the empirical anchor for GAAs: the ecosystem exposes the registry, versioning, social, incident, and response layers simultaneously to document the chain from publication to containment inside one public ecosystem. The result is therefore neither another scanner paper nor another Moltbook paper, but an ecosystem-chain security report grounded in a real and observable GAA crisis.

3 Methodology

Our methodology is organized into four components. ***Automated ecosystem data collection*** gathers and normalizes public evidence across the OpenClaw ecosystem rather than relying on a single registry snapshot. ***Risk extraction and evidence construction*** aligns external risk assessments with versioned skill artifacts and supporting evidence. ***Lifecycle and propagation modeling*** reconstructs how risk evolves across versions, campaigns, and cross-platform circulation. Finally, ***Statistical design and retrospective control analysis*** maps these evidence layers to **RQ1**–**RQ4** while keeping every claim evidence-bounded. This design follows the same narrative backbone introduced before: *publish* → *update* → *propagate* → *expose* → *contain*.

Table 1 Overview of the sources used in the evidence pipeline.

Source ID	Source Name	Records	Detailed Description
S1	Versioned Skills	48,404	Skills with versioned registry
S2	Moltbook	562,115	Risky posts on Moltbook
S3	GitHub Ecosystem	49,191	Github security events/advisories
S4	Reports + CVEs	5,036	Incident reports and vulnerability disclosures
S5	Community Channels	3,768	Promotion and propagation signals from public channels

3.1 Automated Ecosystem Data Collection

Multi-source collection framework. Registry and version data reveal what is published and how it changes, but they cannot by themselves show incident grounding, propagation, or response. Conversely, incident reports, GitHub traces, and community channels are valuable for context and validation, but they cannot replace version-level registry evidence when the question is how risk emerges or mutates. We therefore collect public and reproducible evidence from five source families (S1-S5), covering the registry, community, incident, and response layers of the ecosystem. Unless explicitly stated otherwise, the evidence-bearing collections used for the current report are frozen through March 18, 2026.

The current evidence bundle is given in Table 1. The largest source family is Moltbook-scale social data (S2, 562,115 records), followed by GitHub ecosystem traces (S3, 49,191), and registry plus version-archive data (S1, 48,404). The remaining sources provide incident anchors and cleaner propagation evidence, including reports plus CVEs (S4, 5,036) and public community channels (S5, 3,768). At the signal level, the current bundle already includes 83 incident reports, 91 CVE disclosures, 11,539 GitHub security events, and 374 high-precision risky Moltbook operations. S1 is the methodological backbone for **RQ1** and **RQ2** because only versioned registry evidence can reveal first-risky release, benign-first-to-risky transitions, and later version changes. S2 and S5 form the propagation backbone for **RQ3**, but with different tradeoffs: S2 provides scale and candidate linkage, whereas S5 provides a smaller but cleaner signal set. S3 and S4 provide response-side and incident-side triangulation.

Normalization and analysis-ready views. We implement the collection pipeline as an automated, provenance-first workflow. Source-specific collectors ingest raw artifacts from the five source families, after which the framework performs de-duplication, relevance filtering, timestamp normalization, schema normalization, and canonical identifier resolution. The goal is not to flatten everything into a single table. Instead, we construct analysis-ready views that preserve the strengths of each source while making cross-source joins explicit and auditable. These derived views include a skill-version table for lifecycle analysis, incident-anchor tables for campaign reconstruction, community-linkage candidates with explicit confidence levels, and GitHub response tables for release-side containment analysis. This separation is necessary because descriptive inventory counts and report-level conclusions are not the same object. The pipeline therefore keeps raw collection, normalized evidence, and paper-level claims as distinct layers. By keeping the pipeline in this order, the framework can support both large-scale collection and evidence-bounded inference within the same analysis design.

3.2 Risk Extraction and Evidence Construction

External risk assessment as the primary risk signal. The first analytical stage aligns external risk assessments with versioned skill artifacts. The goal is not to turn the paper into another generic scanner study Liu et al. (2026), but to build reproducible evidence about what kinds of risky behavior are present in the OpenClaw ecosystem and when they first appear. For this reason, the primary analytical unit is the *skill-version* rather than the latest visible state of a skill. Latest-state summaries can hide whether risk existed from first publication, emerged later through an update, or oscillated across repeated releases.

Because there is still no unified, authoritative, and practically usable benchmark that can deterministically judge the security status of in-the-wild agent skills, we do not treat a self-defined taxonomy as the final risk oracle. Instead, we use the risk assessments already exposed by the two ecosystem-facing services that currently provide operational skill vetting. On the ClawHub side, we use the platform's own OpenClaw risk

Table 2 Version-derived metrics used for lifecycle analysis.

Metric	Unit	Definition
First Risky Version Index	*Skill*	Order of the earliest version marked risky by external assessment or corroborated evidence
Time-to-Poisoning	*Days*	Time from first benign version to first risky version
Suspicious Streak Length	*Skill*	Number of consecutive risky versions in the version history
Risk-state Flip Count	*Skill*	Number of transitions between risky and non-risky states across versions
Post-flag Churn	*Skill*	Release cadence after the first risky version, used as a post-flag update proxy

signals together with VirusTotal-backed checks OpenClaw (2026c); VirusTotal (2026). On the `skill.sh` side, we use the assessments surfaced through Gen Agent Trust Hub Gen Digital (2026), Socket Socket (2026), and Snyk Snyk (2026). These external assessments are treated as the primary risk results for each skill or version. Our local artifact extraction layer is used only to align those results with version history, explain later transitions, and support campaign- or lifecycle-level interpretation.

Evidence triangulation and confidence control. External assessment alone is not sufficient for strong ecosystem-security claims. We therefore use provenance-rich triangulation to decide which observations can support descriptive results, campaign attribution, or propagation findings. Each candidate observation may be supported by one or more of the following: incident reports, report-side observables, registry-side risk warnings, GitHub advisories or release activity, and community linkage clues. No single weak source is allowed to stand in for a stronger ecosystem claim.

This evidence matrix is deliberately asymmetric. Incident reports are strong anchors, but they still need artifact-level corroboration when the paper makes version-level or campaign-level claims. Registry-side warnings can support prioritization, but do not by themselves imply malicious intent. GitHub is useful for measuring release-side response and containment timing, yet it is not complete attack-side ground truth. Community linkage can strengthen a claim, but only after confidence-aware mapping. For all headline findings, the paper therefore prioritizes externally observed risk assessments plus cross-source corroboration over internally defined heuristics.

3.3 Lifecycle and Propagation Modeling

Version-aware lifecycle reconstruction. The second stage models how risk evolves over time. Starting from the S1 backbone, we reconstruct an ordered version history for each skill using registry version identifiers and release timestamps. This allows us to move from the descriptive question "what is risky" to the lifecycle question "when did risk first appear," which is the core methodological requirement behind **RQ2**. From this history, we derive version-aware lifecycle metrics such as the first risky version index, time-to-poisoning, suspicious streak length, risk-state flip count, and post-flag churn. These metrics distinguish between skills that are risky from first release and skills that become risky only after trust, installs, or visibility have already accumulated. That distinction is what turns the paper from a static registry scan into a lifecycle-focused ecosystem analysis. Version order is determined from release timestamps and registry version identifiers, and the first risky version is the earliest version whose external risk assessment or corroborated evidence marks it as risky.

Campaign reconstruction. Version evolution alone does not explain whether risk is isolated or organized. To address **RQ1** at the ecosystem level, we reconstruct campaigns with deterministic multi-signal clustering rather than embedding-only grouping. Skills are linked into the same campaign only when multiple signals agree, such as publisher reuse, naming templates, synchronized publication or update windows, shared risk signals, or linked report observables. This requirement reflects the paper's evidence standard. Campaign claims must be interpretable and traceable to observable evidence rather than artifacts of one permissive similarity metric. The resulting clusters are later used to quantify concentration, burstiness, and repeated abuse structure in the threat landscape.

Cross-platform propagation linkage. To answer **RQ3**, we treat propagation as a confidence-aware linkage problem across registry and community layers. S2 provides broad Moltbook-scale candidate evidence, while S5 provides a smaller but cleaner propagation set from community channels. The framework does not treat thematic similarity as sufficient evidence of propagation. Instead, it extracts explicit signals such as canonical

Table 3 Threat surface of current OpenClaw skill ecosystem.

Metric	Record Count	Detailed Description
Total Skills	21,420	Distinct skills in the registry snapshot
Total Versions	26,984	Released versions in the version archive
Versions Per Skill	Median=1; P_{95}=5	Version history depth
Downloads Per Skill	Median=265; P_{95}=2583	Heavy-tailed skill popularity
Ever-Risky Skill Share	9,837 / 21,420 (45.92%)	Skills with at least one risky version at any point
Latest-Risky Skill Share	9,435 / 21,420 (44.05%)	Skills whose latest observed version is risky
Incident Baseline Count	83	Public incident and investigation reports
CVE Baseline Count	91	NVD/CVE records related to the ecosystem

skill URLs, install commands, exact slugs, executable patterns, and credential-related terms, and then assigns a linkage confidence level before any downstream analysis is performed. The linkage rubric has three levels. Low-confidence linkage captures thematic overlap only and is excluded from headline claims. Medium-confidence linkage captures behavior-oriented or skill-like references whose mapping remains ambiguous and is used only as supporting evidence. High-confidence linkage requires an exact artifact reference, such as a canonical URL, install command, or exact slug, and is the only class allowed to support the strongest propagation findings. This structure prevents broad community chatter from being inflated into causal-security claims while still allowing the method to analyze propagation at ecosystem scale. In practice, a Moltbook post containing a canonical skill URL or an explicit `claw install <slug>` command is scored as high-confidence linkage, whereas a fuzzy mention of a skill-like name without a resolvable artifact is treated as low-confidence only.

3.4 Statistical Design and Retrospective Intervention Analysis

RQ-aligned statistical design. The statistical layer follows the same evidence-bounded logic as the collection and alignment stages. For **RQ1**, the main analytical objects are skills, versions, publishers, and campaign clusters; we use baseline summaries, heavy-tail distribution views, Lorenz curves, campaign-size distributions, and pattern-family counts to test whether risk is structurally concentrated rather than uniform. For **RQ2**, the unit of analysis is the ordered skill-version timeline; we use lifecycle state counts, first-risky version indices, time-to-poisoning summaries, release-cadence comparisons, and aligned incident-response timelines to characterize benign-to-risky transitions, mutation, and delayed containment. For **RQ3**, the unit of analysis is the promotion event or linked artifact pair; we use confidence-stratified signal counts, platform summaries, burstiness statistics, repeated-template counts, and odds-ratio / Fisher tests to evaluate whether higher-confidence propagation signals are selectively coupled to risky artifacts. These methods are chosen for interpretability and compatibility with observational evidence, not for maximal statistical complexity.

Retrospective intervention analysis. **RQ4** converts the previous analyses into practical security implications. Here the method asks not only where risk exists, but where it could have been intercepted. We therefore compare control points retrospectively at publication time, update time, propagation time, and response time. The key output is comparative leverage: which stage covers the largest share of observed risk, under what evidence assumptions, and with what operational limitations. This design keeps the paper away from unsupported causal claims. We do not infer victim impact from downstream visibility, do not treat install-related traces as proof of harm, and do not interpret GitHub-side activity as a complete chronology of attack emergence. Instead, the statistical design ensures that every result is matched to a method whose assumptions are consistent with the evidence actually collected. The purpose of the section is therefore not only to analyze the ecosystem, but also to bound what the paper is and is not justified to claim. GitHub is treated as primary evidence only for response-side timing and release hardening, and as supporting context elsewhere.

4 Threat Landscape of the OpenClaw Ecosystem

RQ1 seeks to answer: what is the OpenClaw threat landscape? Our goal is not merely to count risky artifacts, but to determine whether ecosystem risk is large, layered, and structurally concentrated rather than random.

Table 4 Representative campaign evidence and clustering signals.

Campaign	Size	Evidence Signals
Signal Cluster: Credential Exposure	5,326	Credential exposure, persistence, and metadata mismatch; linked to 31 report records with medium confidence.
Publisher Cluster: ivangdavila	311	Repeated persistence, credential-exposure, and metadata-mismatch signals concentrated under one publisher cluster.
Campaign Term: Prompt Injection	48	Prompt injection, credential exposure, and persistence; linked to 20 report records with high confidence.

To answer this question, we combine the registry and version backbone with incident, vulnerability, and response evidence so that the threat landscape is analyzed across the ecosystem rather than from one artifact family alone.

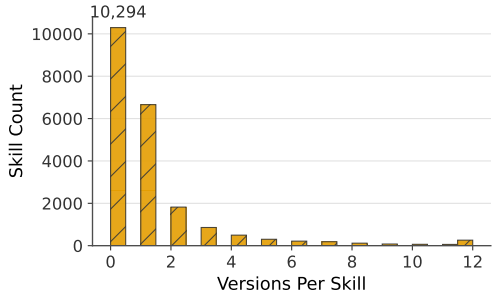


Figure 2 Distribution of version counts per skill.

Ecosystem-wide risk baseline. We begin with the broadest possible question: how large is the observable OpenClaw risk surface? Table 3 shows that the current snapshot contains 21,420 distinct skills and 26,984 observed versions, indicating that the ecosystem is already large and version-active. Figure 2 shows that this version history is not cosmetic: although the median skill has only one version, the 95th percentile reaches five versions, so a meaningful long tail of skills continues to evolve after first publication. Popularity is similarly skewed. As shown in Figure 3, downloads per skill follow a heavy-tailed distribution with a median of 265 and a 95th percentile of 2,583, meaning that visibility and adoption are concentrated in a relatively small subset of skills.

The same baseline also shows why a latest-state view is insufficient. Table 3 reports that 9,837 of the 21,420 skills (45.92%) are *ever risky*, while 9,435 (44.05%) are risky in their latest observed version. The percentage gap is modest, but its meaning is not: counting only the latest visible state hides skills whose risky behavior appeared and later changed. Beyond the registry itself, the ecosystem also contains 83 public incident reports and 91 CVE disclosures, showing that the threat surface spans both registry artifacts and external security reporting.

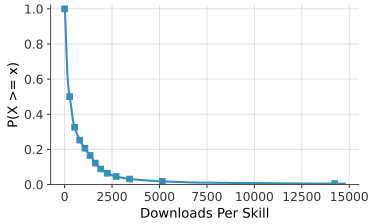


Figure 3 Distribution of skill popularity in the OpenClaw registry.

Risk concentration across publishers and campaigns. Scale alone does not explain ecosystem risk. The next question is whether risky burden is evenly distributed or concentrated in a smaller set of publishers and reusable abuse structures. Our results support the second interpretation. Figure 4 shows a strong deviation from the equality line, indicating that risky burden is concentrated across publishers rather than uniformly distributed.

Campaign reconstruction sharpens the same point. The framework identifies 621 campaign clusters across the risky-skill set, and Figure 5a shows an extremely steep size distribution: the largest cluster contains 5,326 risky skills, while the next largest drops to 311. Table 4 makes this structure concrete by showing that the dominant clusters are not random collections of skills, but repeated combinations of credential exposure, persistence, metadata mismatch, and prompt-injection signals aligned with external reports.

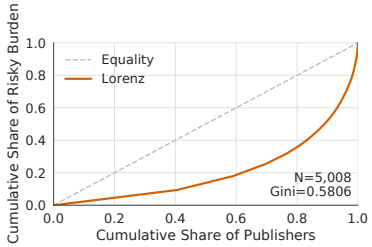
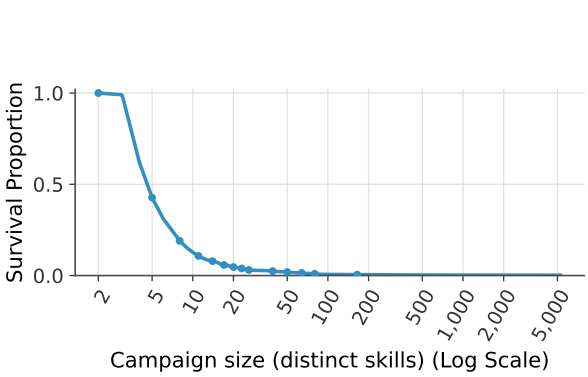


Figure 4 Concentration of risky burden across publishers.

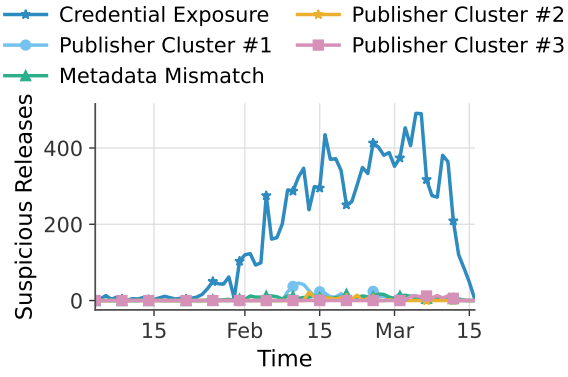
This concentration is stronger at the campaign level than at the single-publisher level. The top campaigns account for 60.77% of all risky skills, while the most influential attributed publisher contributes 3.17% of risky skills on its own. Figure 5b further shows that these clusters are temporally structured rather than static, with repeated bursts among the largest campaigns. Taken together, these results indicate that the threat landscape is concentrated, but not reducible to one dominant publisher. Instead, it is shaped by a mixture of publisher reuse, signal clusters, and recurring abuse templates.

Table 5 Summary of inherited versus OpenClaw-specific risk patterns.

Pattern family	Count	Interpretation
Inherited Instruction Patterns	21,777	Dominant family, driven by credential solicitation and document-code mismatch
Inherited Code Patterns	21,378	Traditional execution, network transfer, and dependency-chain abuse
OpenClaw-specific Patterns	7,823	Risk coupled to OpenClaw-specific capability surfaces, e.g., workspace, memory, MCP, hooks, and runtime configuration



(a) Distribution of campaign sizes.



(b) Temporal activity of the top campaign clusters.

Finding RQ1-1: *The OpenClaw threat landscape is already ecosystem-scale rather than artifact-scale: the registry is large, version-active, and popularity-heavy-tailed, while the same ecosystem simultaneously exhibits incident and CVE evidence beyond the registry itself.*

Attack-surface composition. The final **RQ1** question is what kinds of risk dominate the ecosystem. Here the OpenClaw threat landscape is not limited to one family of abuse. Figure 6 and Table 5 show that inherited instruction patterns are the most prevalent family, matching 21,777 risky versions, followed closely by inherited code patterns at 21,378. OpenClaw-specific patterns remain smaller in total count, but they still appear in 7,823 risky versions, which is too large to dismiss as a corner case.

Finding RQ1-2: *OpenClaw risk is structurally concentrated rather than random: the top campaign clusters cover 60.77% of risky skills, while the largest attributed publisher still contributes only 3.17%, indicating coordinated structure beyond simple one-publisher dominance.*

The segment-level composition in Figure 6 makes this distinction more concrete. Within inherited instruction patterns, *Credential Solicitation* alone appears 21,777 times and *Doc-Code Mismatch* appears 16,836 times. Within inherited code patterns, *Network Transfer Path* appears 20,064 times and *Code Execution Chain* 13,654 times. Among OpenClaw-specific patterns, *Workspace/Memory Surface* and *MCP/Hook Surface* appear 3,910 and 3,870 times, respectively. This means that the ecosystem threat surface is simultaneously shaped by generic skill abuse and by capability surfaces that are specific to how OpenClaw organizes tools, memory, configuration, and hooks.

RQ1 Summary. Taken together, **RQ1** shows that the OpenClaw threat landscape is large, multi-layered, and structurally concentrated. Risk is visible not only in the registry, but also in incident, vulnerability, and response layers. It is organized through campaigns and concentrated publishers rather than random isolated artifacts. And it is shaped by both inherited skill-abuse patterns and OpenClaw-specific capability surfaces, which together justify treating OpenClaw as an ecosystem rather than a single artifact.

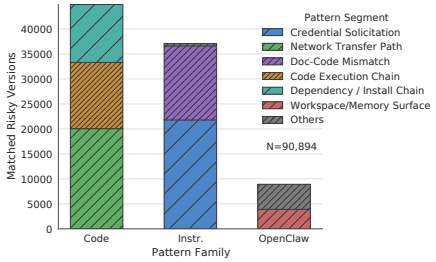


Figure 6 Composition of attack surface across risk-pattern families.

Finding RQ1-3: *The OpenClaw threat surface is both inherited and platform-specific: generic instruction and code abuse dominate in absolute count, but OpenClaw-specific capability surfaces still appear at ecosystem scale and therefore need to be modeled explicitly rather than treated as edge cases.*

5 Risk Evolution Across the Lifecycle

After **RQ1** establishing what the OpenClaw threat landscape looks like, **RQ2** asks how that landscape changes through time. The core question is whether risky skills are dangerous from first release or become dangerous later through updates, mutation, and delayed containment. To answer this question, we analyze the filtered longitudinal cohort of 18,682 skills for which ordered version histories and lifecycle labels can be reconstructed.

From first release to first risky version. The first lifecycle question is where risk first appears. Our results show that among the 18,682 skills in the version-ordered cohort, 9,607 (51.42%) remain never risky across the observation window, 9,040 (48.39%) are risky at birth, and only 35 (0.19%) are benign at first release but become risky later. Although this suggests that publish-time review dominates the defense problem, the later-risky subset remains strategically important because it represents exactly the cases that would be invisible to snapshot-only inspection. Figure 7 shows that these late transitions happen early in the version history rather than after a long benign period. Among the 35 benign-then-risky skills, 27 first become risky at version 2, five at version 3, and three at version 4, with 77.1% of the later-risky subset crosses the threshold at the very next release. This means that update-time review is not primarily about very old skills drifting gradually over dozens of releases. It is about catching fast transitions that occur shortly after trust and visibility are established.

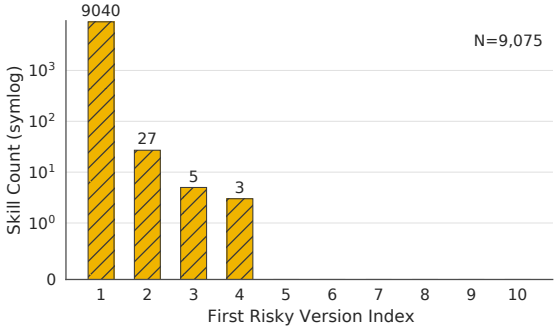
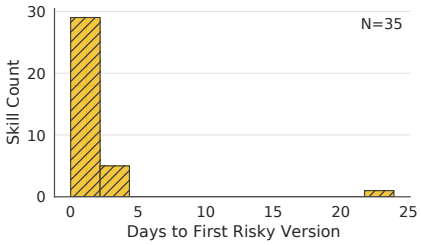
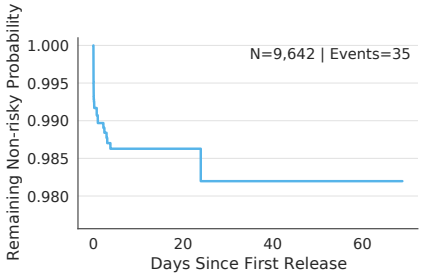


Figure 7 Distribution of the first risky version index.

Finding RQ2-1: *Most observed risky skills are risky from their first visible version, but the benign-then-risky subset exists and transitions early, which makes update-time review a narrow but essential complement to publish-time review.*



(a) Distribution of time-to-poisoning across benign-then-risky skills.



(b) Persistence of benign state over the observed skill lifetime.

Time-to-poisoning and persistence of benign state. The next question is how quickly later risk emerges. Figure 8a shows that among the 35 benign-then-risky skills for which duration is measurable, the median time-to-poisoning is 0.0397 days, or roughly 57 minutes, and the 90th percentile is 2.73 days. This distribution indicates that later-emerging risk usually appears quickly after first release rather than after a long benign maturation period. Figure 8b tells the same story from the opposite direction. The benign-start cohort contains 9,642 skills, but only 35 lifecycle transitions from benign to risky, which corresponds to an event rate of 0.36%. In other words, most benign-first skills remain benign throughout the observed window, while the small subset that does become risky tends to do so quickly. This combination of rarity and speed is what makes poisoning events strategically important even when their absolute count is small.

Mutation, churn, and post-flag evolution. Risk evolution doesn't stop once a skill first becomes risky. The next lifecycle question is whether risky skills exhibit different update behavior after surfacing. Here the data point to a subtler pattern than simple burstiness. Figure 9 shows that the median per-skill median inter-release interval is 0.0686 days for never-risky skills, 0.0650 for risky-at-birth skills, and 0.0658 for benign-then-risky skills, with broadly similar 90th-percentile intervals across groups.

This similarity is itself informative. It suggests that later-risky skills do not necessarily reveal themselves through obviously abnormal release cadence alone. In practice, this means that update timing is not a strong enough signal by itself; the content of the version diff remains the decisive unit for identifying poisoning, mutation, and suspicious churn. Version-aware review is therefore necessary not because later-risky skills update much faster, but because the risky transition is embedded in otherwise ordinary release activity.

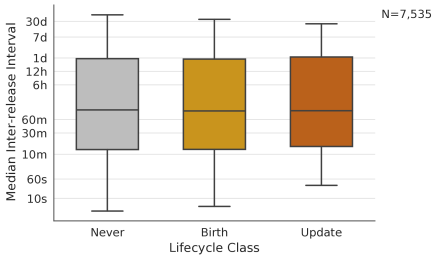


Figure 9 Release-cadence comparison across different lifecycle-risk patterns.

From public signal to observable containment. The final lifecycle stage is containment. Here the method aligns incident reports, CVE disclosures, and GitHub hardening activity on a shared timeline rather than pretending that every risky skill can already be mapped to a precise takedown delay. Figure 10 shows that OpenClaw containment is observable as a downstream stage: public signal accumulates first, and security-oriented hardening appears afterward in release and response artifacts. This timing pattern matters even when per-skill delay estimates remain sparse. It means the report can analyze containment as an ecosystem process rather than only as a policy promise. At the current stage, GitHub remains strongest as response-side evidence, not as a complete attack-side chronology. That boundary is important, but it does not weaken the main lifecycle conclusion. Containment is visible, delayed, and heterogeneous across the same ecosystem in which risk first emerged.

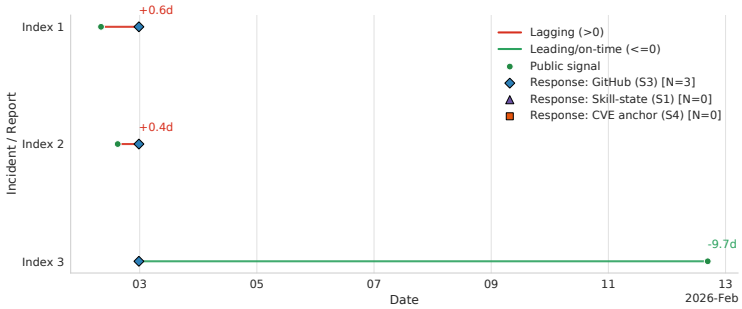


Figure 10 Aligned timeline of public signals and containment activity.

RQ2 Summary. RQ2 shows that OpenClaw risk is lifecycle-structured rather than static. Most risky skills are already risky at first release, but a small benign-then-risky subset exists and transitions quickly after publication. These later transitions do not separate cleanly by update cadence alone, which makes version-diff analysis essential. And once public signal appears, containment can be observed as a delayed ecosystem response rather than treated as a black box.

> ***Finding RQ2-3:*** *Containment is an observable downstream stage rather than an assumed one: the ecosystem exposes public signal first and response artifacts later, even though direct skill-level takedown delays remain incomplete.*

6 Risk Propagation Across the Ecosystem

RQ3 asks whether OpenClaw risk remains confined to the registry or moves across the broader ecosystem. Our concern here is not generic social chatter. It is whether risky artifacts are referenced, promoted, duplicated, and aligned with other ecosystem layers strongly enough to matter for ecosystem security analysis. To answer this question, we analyze propagation events with explicit confidence levels and then test whether the higher-confidence subset is coupled to risky artifacts.

Propagation signal inventory. The first step is to establish what the observable propagation surface looks like. Across the current bundle, we identify 1,884 promotion events. Most of these are weak signals rather than direct artifact references: 1,520 events (80.68%) are low-confidence thematic mentions, 318 (16.88%) are medium-confidence behavior signals, and only 46 (2.44%) are high-confidence links. Within the high-confidence subset, 43 events are command-based links and three are direct URL-based links. The basis and deduplicated taxonomy used to extract these Moltbook

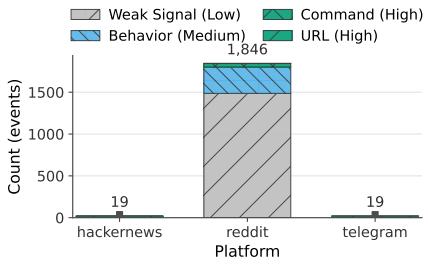


Figure 11 Composition of propagation signals across platforms and confidence.

Table 6 Summary of propagation-event signal types.

Signal Category	Count	Share	Interpretation
Weak Signals	1,520	80.68%	Thematic overlap only; excluded from headline claims.
Behavior Signals	318	16.88%	Medium-confidence behavior cues that may support propagation analysis.
Command Signals	43	2.28%	High-confidence linkage through explicit install or execution signals.
URL Signals	3	0.16%	High-confidence linkage through exact skill or artifact URLs.

and community-side patterns are summarized in Appendix A, especially Table 8. Figure 11 and Table 6 make this split explicit: weak thematic mentions dominate the observable surface, while the actionably linked subset is small by design. The platform distribution is even more skewed. Reddit contributes 1,846 events, while Hacker News and Telegram contribute 19 each. This does not mean Reddit is inherently more dangerous. It means that the currently observable propagation surface is dominated by one platform, and the rest of the ecosystem remains much sparser under the present collection window.

> ***Finding RQ3-1:*** *The observable propagation surface is real but highly uneven: most events are weak thematic mentions, and only 46 of 1,884 promotion events qualify as high-confidence artifact links.*

Cross-platform structure and repeated propagation patterns. Propagation matters most when it is structured rather than random. Here the evidence again points to organization rather than noise. Figure 12 shows that the broader propagation corpus is highly bursty rather than smooth, which is consistent with the summary statistics of 21,254 duplicate posts, a burstiness Fano factor of 98.79, and a peak-share statistic of 12.53%. The split between the *risky-linked* and *generic* series also shows that risky-linked activity does not simply mirror background chatter; it rises sharply in the late-February and early-March window where the broader social volume also accelerates.

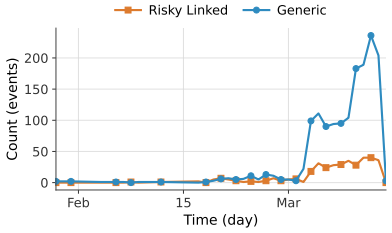


Figure 12 Temporal burstiness of propagation events across the ecosystem.

Figure 13 reveals what these bursts look like in practice. The largest template class is *General security discussion* with 199 events, followed by *Verification and testing guidance* with 32, *Security audit findings* with 31, and *Skill installation instruction* with 28. The presence of installation-oriented and security-audit templates among the most repeated patterns matters more than their raw count. It shows that propagation is not limited to abstract discussion; some repeated events include operational instructions and artifact-facing guidance that could plausibly move risky skills through the ecosystem.

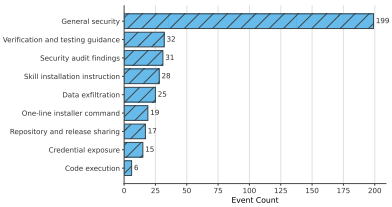


Figure 13 Distribution of repeated propagation templates.

Figure 14 adds one more layer of evidence. Among 374 high-precision risky operations extracted from Moltbook-scale traces, the dominant standardized class is *Execution* with 178 events, far exceeding *Initial Access* with 28, while 168 remain uncategorized or mixed. This class composition is consistent with a propagation layer in which the most visible operational signals are already close to execution rather than merely high-level discussion. Platform-level structure points in the same direction. Table 7 shows that Reddit contributes 45 high-confidence events and a 19.39% risky-signal share, Hacker News contributes one high-confidence event and a 21.05% risky-signal share, and Telegram contributes no high-confidence events in the present run. The sample outside Reddit is small, but the result still supports a structured rather than uniform picture of propagation.

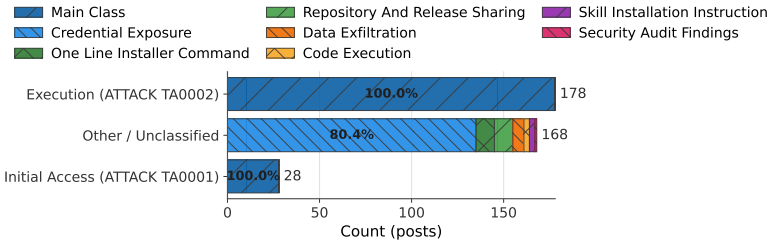


Figure 14 Distribution of risky-operation classes in the propagation layer.

Table 7 Platform summary of high-confidence propagation signals.

Platform	Events	High-Conf. Events
Reddit	1,846	45
Hacker News	19	1
Telegram	19	0

Propagation-risk association. Structure alone does not prove security relevance. The stronger question is whether linked or promoted artifacts are more likely to be risky than the rest of the ecosystem. At the current evidence level, the answer is suggestive but still evidence-bounded. Figure 15 show that among 11,126 slugs in the comparison set, only 26 belong to the linked subset, but all 26 are risky, whereas the unlinked risky share is 88.24%. This yields an odds ratio of 7.06 for linked versus unlinked risky skills, with a wide confidence interval and a Fisher exact p-value of 0.065.

The same pattern appears in the install-command linked subset and the Reddit-linked subset: both have 100% risky share and an odds ratio of 4.39, again under wide uncertainty because the sample is small. By contrast, the execution-pattern and credential-term subsets each contain only two linked slugs; both remain 100% risky in observed share, but their odds ratios drop to 0.66 with extremely wide confidence intervals, indicating that these micro-subsets are too small for stable inference. In other words, the current data do not yet justify a strong causal claim, but they do show that the higher-confidence propagation subset is structurally more security-relevant than generic chatter, while also making clear where the sample is still too thin to support stronger conclusions.

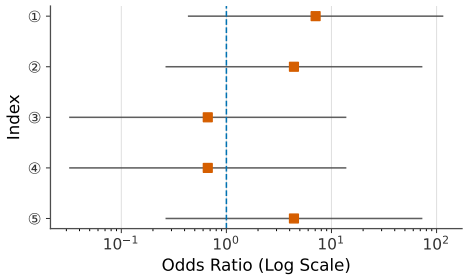


Figure 15 Association between confidence-aware propagation subsets and risky artifacts. ①, ②, ③, ④ and ⑤ denote linked vs unlinked risky skills, install-command linked subset, execution-pattern linked subset, credential-term linked subset and reddit platform linked subset, respectively.

> ***Finding RQ3-2:*** *Propagation events are highly bursty and template-driven rather than randomly scattered, and the currently observable high-confidence propagation surface is dominated by Reddit.*

Propagation in relation to downstream visibility. The strongest version of **RQ3** asks whether propagation connects to the broader ecosystem chain rather than stopping at social visibility. The current downstream alignment stage places 1,884 promotion events, 1,084 linkage events, and 83 incident records on the same temporal frame. Figure 16 shows that promotion activity rises sharply in late February and early March, while incident lanes remain visible in the same period. This does not prove that promotion causes later compromise. What it does show is that propagation belongs on the same ecosystem timeline as incident visibility and public security reporting. That structural coupling is enough to justify treating propagation as part of the ecosystem risk chain rather than as a side channel of attention.

RQ3 Summary. **RQ3** shows that OpenClaw risk does move beyond the registry, but that this movement must be studied under confidence-aware rules. Most observable propagation events are weak thematic signals, yet the smaller high-confidence subset is disproportionately risky. Propagation is also structured through bursts, repeated templates, and platform concentration rather than random discussion. Taken together, these results justify keeping propagation as a first-class ecosystem layer while preserving a cautious boundary against unsupported causal claims.

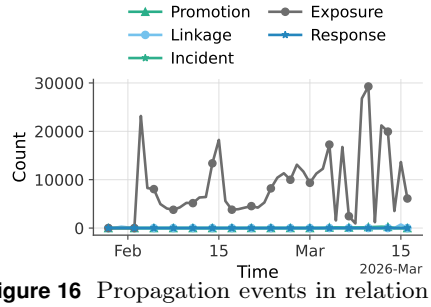


Figure 16 Propagation events in relation to downstream incident visibility.

> ***Finding RQ3-3:*** *High-confidence propagation signals are selectively coupled to risky artifacts, but the current linked subset remains small enough that the result should be interpreted as structural evidence rather than causal proof.*

7 Defense Leverage Points

RQ4 converts the previous analyses into practical control-point reasoning. If **RQ1** maps the threat landscape, **RQ2** explains lifecycle evolution, and **RQ3** explains propagation, then the remaining question is where defenders can intervene most effectively along the same chain. Our goal is not to claim that one stage solves the entire

problem. It is to compare the leverage of publish-time, update-time, propagation-time, and response-time controls under the evidence currently available.

Publish-time controls. The strongest argument for publish-time controls comes from the lifecycle results. Within the 9,075 eventually risky skills in the version-ordered cohort, 9,040 are already risky at first release. Under the current labeling and static-analysis rules, this means that publish-time review covers 99.61% of the eventually risky skills in the longitudinal cohort. This result should not be overstated. It does *not* mean publish-time screening is sufficient for the full ecosystem. It means that broad first-release review remains the highest-coverage intervention point under the currently observed data. In practice, this makes registry-side static analysis and first-release risk extraction the most natural first barrier, especially for high-volume threats that already reveal themselves at publication time.

Update-time controls. Update-time controls matter for a different reason. Only 35 skills in the longitudinal cohort are benign at first release and risky later, but those 35 cases are exactly the ones that snapshot-only review will miss. Moreover, the first risky transition happens quickly: 27 of the 35 poisoning candidates become risky at version 2, five at version 3, and three at version 4. This means update-time review is low-volume but strategically unique. The implication is straightforward. If defenders remove update-time diff inspection from the chain, they lose visibility into the trust-then-poison pattern entirely. Conversely, if they add targeted update-time checks, the incremental operational burden is limited because the number of later-risky cases is small. Update review therefore offers narrow but irreplaceable coverage.

Propagation-time controls. Propagation-time controls operate on a narrower surface than publish-time review, but potentially with higher precision. Among 1,884 promotion events, only 46 qualify as high-confidence artifact links, and 43 of those are explicit command-based links. This means the actionable propagation subset is small, but it is also operationally concrete: install commands, exact URLs, and artifact-resolvable skill references are all intervention points that a platform or community operator can monitor directly. The association results support the same conclusion. The high-confidence linked subset is disproportionately risky, even though the current sample is too small for strong statistical certainty. In practical terms, propagation-time controls are unlikely to deliver the widest coverage, but they can help suppress visibility and downstream diffusion for a subset of risky artifacts that are already strongly tied to operational promotion behavior.

Response-time controls. The final intervention point sits downstream of publication, updates, and propagation. Here the clearest observable evidence comes from public response and release activity rather than from a dedicated deployment-side source. GitHub advisories, issues, commits, and security-oriented release notes show when maintainers begin to react to public signal and which parts of the platform they harden in response. These controls do not prevent risky skills from being published or promoted. Instead, they determine how quickly ecosystem risk becomes visible to defenders and how quickly public hardening work begins after that signal appears. Response-time controls therefore have a different role from registry-side review. They are not the first barrier in the chain, but they are the clearest observable barrier once upstream stages have already failed. This makes them complementary rather than substitutive: they reduce residual risk by shrinking the window between public signal and defensive reaction.

> ***Finding RQ4:*** *No single intervention stage is sufficient, but the control-point tradeoff is already clear. Publish-time review offers the broadest current coverage. Update-time review is indispensable for later-risky transitions even though that subset is small. Propagation-time controls are narrow but operationally precise. And response-time hardening remains essential because it is the clearest observable downstream reaction once upstream failures have already occurred.*

RQ4 Summary. RQ4 turns the previous three sections into an actionable ecosystem conclusion. OpenClaw does not need one perfect defense. It needs layered controls aligned with the same chain through which risk is produced and amplified. The evidence therefore favors a defense strategy that combines broad publish-time screening, targeted update-time review, confidence-aware propagation controls, and faster response-time hardening after public signal appears.

8 Discussion and Limitations

8.1 Discussion

The OpenClaw manuscript becomes non-incremental only if it keeps the chain intact. A paper that stops at skill detection would collide with prior marketplace-scanning work Liu et al. (2026). A paper that stops at Moltbook discourse would collide with prior agent-social-network characterization Manik and Wang (2026); Chen et al. (2026). This manuscript avoids both traps by making one narrower but stronger claim. The research object is an ecosystem failure process, not a bag of risky artifacts. That design choice already shapes the paper's evidence. Incident reports and registry artifacts justify threat-landscape and campaign framing. Version history justifies lifecycle evolution. Community traces justify retaining propagation as an observable layer without overstating it. Release-side artifacts justify downstream defense-leverage analysis once public signal appears. Each artifact family is included because it serves the end-to-end story.

8.2 Limitations

The largest current limitation is the absence of a completed longitudinal registry corpus. Without that corpus, we cannot yet report registry-wide malicious-skill prevalence, version-transition rates, or publisher concentration with final confidence. The manuscript therefore treats those results as explicit future computations rather than hidden assumptions. The second limitation is propagation-linkage sparsity. The current community bundle proves collection feasibility but not yet a large high-confidence linkage set between promoted content and concrete risky skills. This means that **RQ3** must remain confidence-aware and evidence-bounded. The third limitation is label incompleteness. External reports provide strong seeds, but they are not a substitute for artifact-level validation. The final paper must still supply manual review, provenance-backed rationales, and adjudication support before publishing maliciousness statistics. The fourth limitation is that several currently collected artifacts are demos. We follow the user's instruction and the repository's quality rules by refusing to convert those demos into final report claims. This limitation weakens superficial completeness. It strengthens the paper's credibility.

9 Conclusion

This report documents OpenClaw as an ecosystem security problem rather than as a collection of isolated bugs or isolated malicious skills. By aligning versioned registry artifacts with incident reports, CVE records, GitHub response activity, and confidence-aware community traces, the report follows the same chain through which risk is published, updated, propagated, exposed, and eventually contained. That ecosystem-chain framing is the main organizing conclusion of the manuscript. The current evidence already supports several concrete takeaways. OpenClaw's threat surface is large, multi-layered, and structurally concentrated rather than random. Most eventually risky skills are already risky at first release, but a small benign-then-risky subset exists and transitions quickly enough that update-time review cannot be removed from the defense story. Propagation beyond the registry is observable, but the actionably linked subset remains small, which means public promotion should currently be interpreted as structural ecosystem evidence rather than as causal proof of downstream compromise. And once public signal appears, containment becomes visible through disclosure and release-hardening artifacts, making downstream response a measurable part of the same chain. These observations imply a practical defense posture rather than a single headline fix. Broad publish-time screening still offers the widest current coverage. Targeted update-time review remains necessary for trust-then-poison cases. Confidence-aware propagation monitoring can help suppress a narrower but operationally meaningful subset of risky diffusion. And response-time hardening remains essential because it is the clearest observable barrier after upstream controls have already failed. At the same time, this report intentionally stops short of stronger claims than the evidence can support. It does not claim final registry-wide malicious-skill prevalence, complete publisher attribution, or causal impact from social amplification. The most important next step is therefore to replace the present evidence bundle with a frozen longitudinal registry corpus, denser high-confidence propagation linkage, and manual adjudication support for maliciousness statistics. If those pieces are added, the current report can serve as the evidence policy, analytical scaffold, and baseline security picture for a fuller ecosystem study later on.

References

Varun Pratap Bhardwaj. Skillfortify: Formal analysis and supply chain security for agentic ai skills, 2026. https://zenodo.org/doi/10.5281/zenodo.18787663.

Eason Chen, Ce Guan, Ahmed Elshafiey, Zhonghao Zhao, Joshua Zekeri, Afeez Edeifo Shaibu, Emmanuel Osadebe Prince, and Cyuan Jhen Wu. Openclaw ai agents as informal learners at moltbook: Characterizing an emergent learning community at scale, 2026. https://arxiv.org/abs/2602.18832.

Xinhao Deng, Yixiang Zhang, Jiaqing Wu, Jiaqi Bai, Sibo Yi, Zhuoheng Zou, Yue Xiao, Rennai Qiu, Jianan Ma, Jialuo Chen, Xiaohu Du, Xiaofang Yang, Shiwen Cui, Changhua Meng, Weiqiang Wang, Jiaxing Song, Ke Xu, and Qi Li. Taming openclaw: Security analysis and mitigation of autonomous llm agent threats, 2026. https://arxiv.org/abs/2603.11619.

Ben Dong, Hui Feng, and Qian Wang. Clawdrain: Exploiting tool-calling chains for stealthy token exhaustion in openclaw agents, 2026. https://arxiv.org/abs/2603.00902.

Gen Digital. Gen launches agent trust hub for safer agentic era. https://investor.gendigital.com/news/news-details/2026/Gen-Launches-Agent-Trust-Hub-for-Safer-Agentic-Era/default.aspx, 2026.

Xinyi Hou, Yanjie Zhao, and Haoyu Wang. On the (in)security of llm app stores. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 317–335, 2025. doi: 10.1109/SP61157.2025.00117.

Xiaojun Jia, Jie Liao, Simeng Qin, Jindong Gu, Wenqi Ren, Xiaochun Cao, Yang Liu, and Philip Torr. Skillject: Automating stealthy skill-based prompt injection for coding agents with trace-driven closed-loop refinement, 2026. https://arxiv.org/abs/2602.14211.

Koi Security. Clawhavoc: 341 malicious clawed skills found by the bot they were targeting. https://www.koi.ai/blog/clawhavoc-341-malicious-clawedbot-skills-found-by-the-bot-they-were-targeting, 2026.

Yi Liu, Weizhe Wang, Ruitao Feng, Yao Zhang, Guangquan Xu, Gelei Deng, Yuekang Li, and Leo Zhang. Agent skills in the wild: An empirical study of security vulnerabilities at scale, 2026. https://arxiv.org/abs/2601.10338.

Md Motaleb Hossen Manik and Ge Wang. Openclaw agents on moltbook: Risky instruction sharing and norm enforcement in an agent-only social network, 2026. https://arxiv.org/abs/2602.02625.

Moltbook. Moltbook api repository. https://github.com/moltbook/api, 2026.

National Institute of Standards and Technology. National vulnerability database. https://nvd.nist.gov/, 2026.

Marten Oltrogge, Erik Derr, Christian Stransky, Yasemin Acar, Sascha Fahl, Christian Rossow, Giancarlo Pellegrino, Sven Bugiel, and Michael Backes. The rise of the citizen developer: Assessing the security impact of online app generators. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 634–647, 2018. doi: 10.1109/SP.2018.00005.

OpenClaw. Clawhub documentation. https://docs.openclaw.ai/tools/clawhub, 2026a.

OpenClaw. Clawhub http api documentation. https://github.com/openclaw/clawhub/blob/main/docs/http-api.md, 2026b.

OpenClaw. Clawhub: Skill directory for openclaw. https://github.com/openclaw/clawhub, 2026c.

OpenClaw. Clawhub telemetry documentation. https://github.com/openclaw/clawhub/blob/main/docs/telemetry.md, 2026d.

OpenClaw. Openclaw core repository. https://github.com/clawdbot/clawdbot, 2026e.

OpenClaw. Openclaw releases. https://github.com/clawdbot/clawdbot/releases, 2026f.

SecurityScorecard. 135,000 openclaw instances exposed via insecure default configuration. https://securityscorecard.com/research/135000-openclaw-instances-exposed-via-insecure-default-configuration/, 2026.

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, 2026. https://arxiv.org/abs/2603.10387.

Snyk. Snyk finds prompt injection in 36%, 1467 malicious payloads in a toxicskills study of agent skills supply chain compromise. https://snyk.io/blog/toxicskills-malicious-ai-agent-skills-clawhub/, 2026.

Socket. Socket brings supply chain security to skills.sh. https://socket.dev/blog/socket-brings-supply-chain-security-to-skills, 2026.

TrustAIRLab. Humans welcome to observe: A first look at the agent social network moltbook. https://moltbookobserve.github.io/, 2026.

VirusTotal. Virustotal documentation. https://docs.virustotal.com/, 2026.

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, 2026. https://arxiv.org/abs/2602.08412.

Wiz. Moltbook open database exposure. https://www.wiz.io/blog/moltbook-open-database-exposure, 2026.

A Moltbook Pattern Taxonomy

This appendix summarizes the basis and classification used for extracting security-relevant patterns from Moltbook and adjacent community traces. The taxonomy is designed for ecosystem security analysis rather than for generic content moderation. Its purpose is to identify public posts that may contribute to the ecosystem chain studied in the main text, especially installation, execution, persistence, credential access, propagation, and downstream harm.

A.1 Basis for Pattern Extraction

Pattern extraction is grounded in three kinds of sources. The primary basis is external security standards, which provide the main classification vocabulary and protect the paper from using ad hoc or platform-specific labels where an established standard already exists. The second basis is project-side empirical evidence, including Moltbook data, public incident reports, and linked ecosystem traces that show these patterns actually appear in the OpenClaw setting. The third basis is a small project-specific result-state layer used only for archival severity labeling. This final layer is intentionally separated from process-level statistics and never mixed with standard attack categories in the main analysis.

In practice, the standard basis is complemented by public ecosystem evidence. The taxonomy is informed by the Moltbook dataset and API-facing materials, and cross-checked against the report sources already used elsewhere in the paper, including Koi, Snyk, MITRE, SecurityScorecard, Wiz, Endor Labs, Bitdefender, Oasis, VirusTotal, Bitsight, Conscia, Cyera, and Trend Micro. This means the appendix serves two purposes at once: it documents why the pattern labels are legitimate, and it documents why those patterns are relevant to the OpenClaw ecosystem rather than borrowed abstractly from unrelated threat models.

A.2 Deduplicated Pattern Classes

The original Moltbook pattern list contains fine-grained codes `C1`–`C40`. For analysis, these codes are merged into a deduplicated taxonomy of higher-level pattern classes so that the same operational behavior is not counted multiple times under slightly different surface forms. Table 8 reports the merged classes, their standard basis, representative examples, and why they matter to ecosystem analysis.

Two additional normalization rules matter for interpreting this taxonomy. First, older internal references to `OWASP LLM08` are aligned in this appendix to the 2025 OWASP terminology of `LLM06 Excessive Agency`. Second, process-level analyses in the main paper count only standard attack or weakness classes. Result-state labels remain in the appendix because they are useful for case archiving and severity interpretation, but they are intentionally excluded from the main distribution statistics so that standardized process classes and project-specific outcome labels are not conflated.

Table 8 Deduplicated Moltbook risk-pattern classes used in propagation analysis.

Primary class	Merged codes	Behavior family	Representative pattern	Why it matters
ATTACK: TA0001	C01/C04/C08/ C09/C37	Supply-chain introduction	`clawhub install <slug>` package-manager installs `git clone ... && ./install.sh` `postinstall`	Anchors the publish-to-install part of the ecosystem chain by capturing trusted installation flows that introduce risky artifacts.
ATTACK: TA0002	C02/C05/ C38	Direct execution	`curl ... | bash` `wget ... | sh` `eval(...)`	Captures posts that turn remote content directly into execution.
ATTACK: TA0003	C03/C25/C27/ C28/C29	Persistence	cron jobs `authorized_keys` shell startup edits service autostart	Captures mechanisms that keep malicious logic alive after initial execution.
OWASP: LLM06:2025	C06/C17	Excessive agency	chain/payment commands delegated high-impact actions agent-to-agent forwarding	Captures cases where agent autonomy moves risk from suggestion to irreversible action.
OWASP: LLM01:2025	C07/C15/C16/ C18/C39	Prompt and context injection	direct injection hidden HTML/Markdown memory poisoning delayed triggers	Captures LLM-native manipulation channels central to agent ecosystems.
ATTACK: TA0006	C10/C19/C23/C30/ C31/C32/C33/C40	Credential access	`printenv` credential files IMDS, K8s secrets browser tokens, clipboard scraping	Links propagation patterns to downstream takeover and secret exposure.
ATTACK: TA0010	C12/C36	Exfiltration and remote channel setup	`curl -X POST -d @file ...` reverse shells SSH tunnels DNS tunneling	Captures the move from local collection to external transfer or remote control.
ATTACK: TA0040	C11/C13/C20	Impact	destructive file operations miners encryption or ransom workflows	Captures high-impact end-stage behavior.
ATTACK: TA0004	C14/C24/C35	Privilege escalation and host control	privileged containers `sudo` abuse extension hijack `LD_PRELOAD` injection	Captures expansion from limited footholds to broader host control.
ATTACK: TA0042	C21	Resource development / DDoS coordination	botnet recruitment coordinated flood instructions	Captures cases where propagation helps form attack infrastructure.
CWE: CWE-918	C22	SSRF and boundary abuse	internal-service access metadata endpoints host-network tricks	Captures network-boundary abuse better expressed as a weakness class.
ATTACK: TA0005	C34	Defense evasion	`history -c` log truncation cleanup deletions	Captures post-execution cleanup that weakens auditability and delays containment.
RESULT: COMPLETE_TAKEOVER	C26	Result-state label	combined chain of installation execution, escalation, and persistence	Used only for severity archiving and excluded from process-level statistics.