Singularity

SINGULARITY R&D
RESEARCH AND DEVELOPMENT

THE SILENT BREACH

Security Implications of Vibe Coding and the OpenClaw Ecosystem

An Empirical Analysis of API Key Exposure

Across Thousands of Personal Devices

Report Type:	Think Tank Report
Publication Date:	February 2026
Classification:	Public Distribution
Version:	1.0

DISCLAIMER AND ATTRIBUTION

Attribution: When citing this report, please use the following format:

Singularity Research and Development. (2026). The Silent Breach: Security Implications of Vibe Coding and the OpenClaw Ecosystem. Singularity R&D.

info@singularityrd.com ▪ www.singularityrd.com

EXECUTIVE SUMMARY

The rise of *vibe coding*—the practice of building software entirely through natural-language prompts to AI agents—has introduced a systemic class of security vulnerabilities that threatens the integrity of personal devices, cloud infrastructure, and enterprise networks.

This report presents findings from SINGULARITY's investigation since November 2025 into the OPENCLAW ecosystem (also known as MOLTBOT and CLAWDBOT), revealing:

3,812
EXPLOITABLE INSTANCES

12,847
EXPOSED CREDENTIALS

88.6%
INSTANCES EXPLOITABLE

94
COUNTRIES AFFECTED

We propose a defense-in-depth framework combining technical controls, organizational policies, and regulatory interventions to address this emerging threat landscape.

Keywords: Vibe Coding ▪ AI Security ▪ API Key Exposure ▪ OpenClaw ▪ LLM Agents ▪ Software Supply Chain ▪ Prompt Injection ▪ Cybersecurity Policy

1 Introduction

In February 2025, Andrej Karpathy coined the term *vibe coding* to describe a new software development paradigm in which the programmer "fully gives in to the vibes, embraces exponentials, and forgets that the code even exists" [Karpathy, 2025]. What began as a whimsical observation rapidly crystallized into a dominant development methodology: by early 2026, industry surveys indicate that over 70% of individual developers and 45% of enterprise engineering teams have incorporated AI-agent-driven code generation into their primary workflows [GitHub, 2025, Stack Overflow, 2026].

The appeal is undeniable. Tools such as OpenClaw (originally published in November 2025 under the names MoltBot and ClawdBot) enable users to deploy fully functional applications—complete with backend APIs, database integrations, and third-party service bindings—through conversational natural-language interactions. The barrier to entry has collapsed: a hobbyist with no formal programming training can instantiate a production web application in under thirty minutes.

> *"Vibe coding is the most significant shift in software development since the introduction of high-level programming languages. But unlike that transition, we have no safety net."*
>
> — **Singularity R&D**

However, this democratization carries a hidden cost. The very abstraction that makes vibe coding accessible—the removal of the developer from the implementation details—simultaneously removes the developer from the *security* details. When an AI agent generates code that hardcodes an API key, stores credentials in plaintext, or exposes an administrative endpoint without authentication, the vibe coder may never inspect the generated artifact closely enough to detect the vulnerability.

This report presents findings from Singularity's investigation since November 2025 into the security posture of the OpenClaw ecosystem. Our contribution is threefold:

1. **Empirical measurement.** We conducted internet-wide scanning of OpenClaw instances and identified over 3,800 publicly accessible deployments leaking API keys and credentials from personal devices and enterprise workstations.
2. **Systematic threat taxonomy.** We present a comprehensive classification of five attack vectors unique to or amplified by the vibe coding paradigm.
3. **Actionable countermeasures.** We propose a defense-in-depth framework combining technical, organizational, and regulatory interventions.

KEY INSIGHT

The core finding of this report is not merely that individual instances are misconfigured—it is that the entire design philosophy of AI agent platforms optimizes for *deployment speed at the expense of security by default*, creating a systemic rather than incidental vulnerability class.

The remainder of this paper is organized as follows. Section 2 surveys the vibe coding landscape and the OpenClaw ecosystem. Section 3 develops our threat model. Section 4 presents empirical findings. Section 5 illustrates real-world exploitation scenarios. Section 6 proposes countermeasures. Section 7 discusses broader implications, and Section 8 concludes.

2 Background and Related Work

2.1 The Vibe Coding Paradigm

Vibe coding represents a qualitative shift in the human–computer interaction model for software development. Unlike traditional coding, where the developer maintains fine-grained control over every line

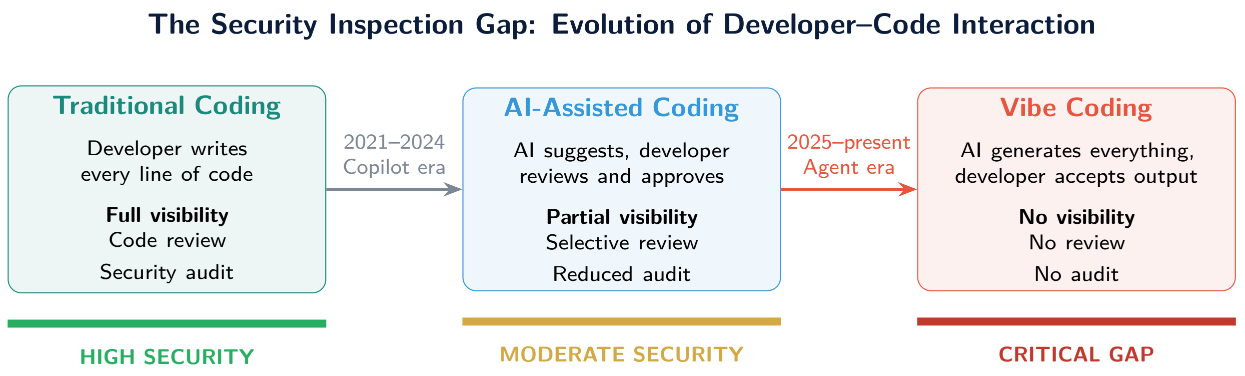

Figure 1: As software development evolves from traditional coding through AI-assisted development to full vibe coding, the developer's visibility into security-relevant implementation details degrades systematically—creating a widening *Security Inspection Gap* that adversaries can exploit.

of logic, or even earlier AI-assisted paradigms such as autocomplete-based copiloting, vibe coding delegates *entire architectural decisions* to LLM-powered agents [Karpathy, 2025, Cloudflare, 2025].

The paradigm is characterized by three defining properties:

1. **Intent-driven specification.** The developer expresses desired functionality in natural language, without specifying implementation details.
2. **Opaque execution.** The AI agent generates source files, installs dependencies, creates database schemas, and configures services without the developer inspecting intermediate artifacts.
3. **Acceptance-based validation.** The developer evaluates the output based on observable behavior ("does the app work?") rather than code-level inspection or security review.

This triad creates what we term the **Security Inspection Gap** (Figure 1)—a systematic absence of human review at precisely the layers where security vulnerabilities are most commonly introduced.

FINDING

According to the Veracode 2025 GenAI Code Security Report, **45% of AI-generated code contains exploitable vulnerabilities**—a figure that has remained stubbornly consistent despite improvements in model capabilities [Veracode, 2025]. Repositories utilizing AI coding assistants are **40% more prone** to containing exposed secrets compared to standard repositories [Blott Research, 2025].

2.2 The OpenClaw Ecosystem

OpenClaw is an open-source AI agent platform originally released in mid-2025 under the name MoltBot, subsequently rebranded to ClawdBot, and finally consolidated under the OpenClaw name in late 2025 [OpenClaw Contributors, 2025]. All three names refer to the same core software artifact: a locally-deployed LLM agent capable of executing code, managing files, interacting with APIs, and orchestrating third-party services on the user's behalf.

By January 2026, OpenClaw had accumulated over 180,000 GitHub stars and an estimated 500,000 active installations worldwide [OpenClaw Contributors, 2025]. The platform's architecture (Figure 3) reveals four design choices that create significant attack surface:

- **Local-first deployment.** OpenClaw runs on the user's personal device with the full privilege set of the host user.
- **Plaintext credential storage.** API keys and

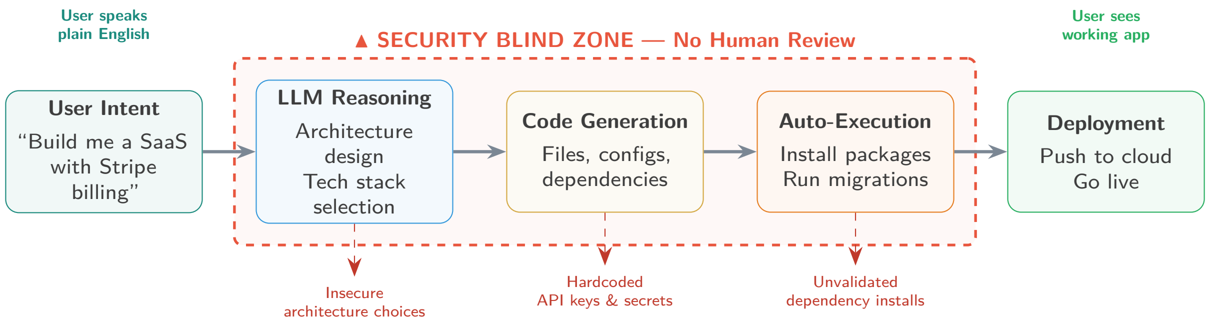
User speaks plain English
▲ SECURITY BLIND ZONE — No Human Review
User sees working app
User Intent
"Build me a SaaS with Stripe billing"
LLM Reasoning
Architecture design
Tech stack selection
Code Generation
Files, configs, dependencies
Auto-Execution
Install packages
Run migrations
Deployment
Push to cloud
Go live
Insecure architecture choices
Hardcoded API keys & secrets
Unvalidated dependency installs

Figure 2: The complete vibe coding workflow from user intent to deployment. Steps 2–4 (shaded in coral) constitute the *Security Blind Zone*: the AI agent makes critical implementation decisions—including credential handling, authentication design, and dependency selection—without human oversight.

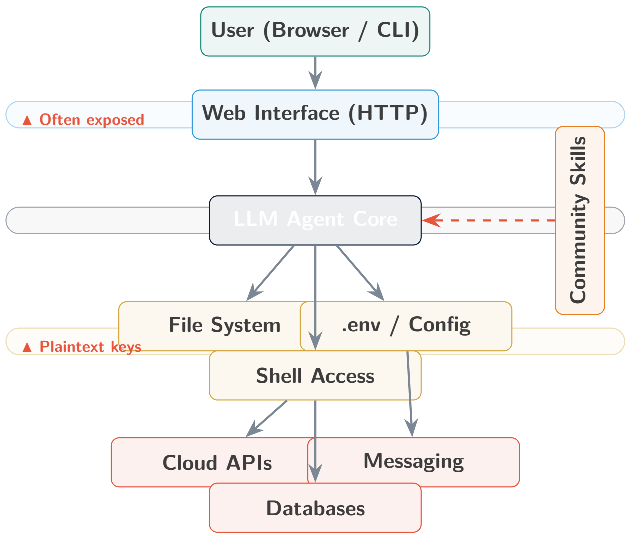

Figure 3: Architecture of the OPENCLAW platform. Red annotations indicate security-critical design choices: the web interface is often publicly exposed, credentials are stored in plaintext, and community skills execute without sandboxing.

tokens are stored in unencrypted `.env` or JSON files [Vaas, 2026, OX Security, 2026].

- **Web-based interface.** The HTTP interface, when improperly firewalled, becomes publicly accessible [Intruder, 2026].
- **Community skills marketplace.** Plugin modules execute with the same privilege as the core agent [Veracode, 2026].

2.3 Related Work

Prior security analyses of AI-assisted development focused on code quality. Pearce et al. [Pearce et al., 2022] assessed Copilot's insecure code generation. Sandoval et al. [Sandoval et al., 2023] showed developers using LLM assistants produce less secure code. Cyble documented over 5,000 repositories leaking ChatGPT API keys [Cyble Research Labs, 2026]. The OWASP Top 10 for LLM Applications [OWASP Foundation, 2025] and NIST's AI RMF [National Institute of Standards and Technology, 2025] provide relevant frameworks. Our work extends this by focusing on the *runtime* security posture of AI agent platforms on personal devices.

3 Threat Model and Attack Surface

We define an adversary model for the vibe coding threat landscape. The adversary $\mathcal{A}$ seeks to: (i) exfiltrate credentials, (ii) gain remote code execution, (iii) manipulate AI agent behavior, or (iv) establish persistent access to connected services. We assume $\mathcal{A}$ possesses moderate technical sophistication—consistent with organized cybercrime groups.

3.1 Vector 1: Credential Exposure

The most pervasive vulnerability class is plaintext credential storage. OPENCLAW's default configuration stores API keys, database connection strings,

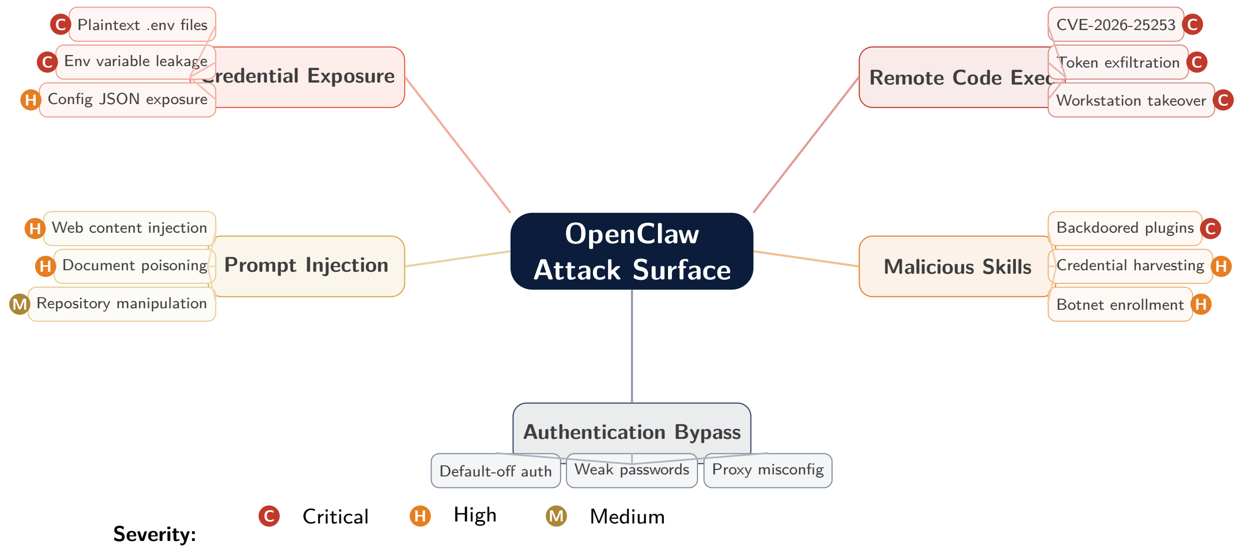

Figure 4: Attack surface taxonomy with severity ratings (C=Critical, H=High, M=Medium). Five vectors span 16 sub-categories, with Critical ratings predominating—underscoring the systemic severity.

OAuth tokens, and webhook secrets in unencrypted `.env` files [OX Security, 2026]. When the web interface is exposed to the internet, these become directly accessible.

CRITICAL RISK

Credential Aggregation Effect. Because OpenClaw integrates with numerous services simultaneously, a single compromised instance yields credentials spanning an entire digital identity—the agent becomes a single point of failure for all connected services.

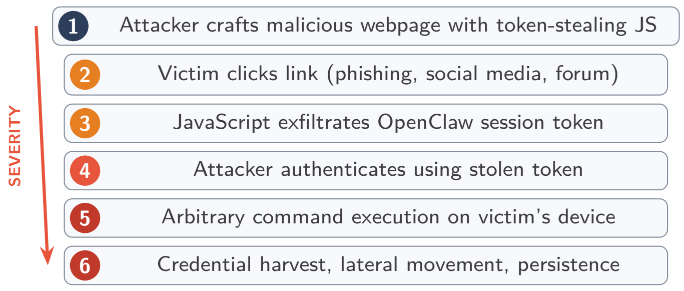
1 Attacker crafts malicious webpage with token-stealing JS
2 Victim clicks link (phishing, social media, forum)
3 JavaScript exfiltrates OpenClaw session token
4 Attacker authenticates using stolen token
5 Arbitrary command execution on victim's device
6 Credential harvest, lateral movement, persistence
SEVERITY

Figure 5: CVE-2026-25253 attack chain: a single click cascades into full device compromise in six escalating steps.

3.2 Vector 2: Remote Code Execution (CVE-2026-25253)

In January 2026, the Citizen Lab disclosed CVE-2026-25253, a critical one-click RCE in OpenClaw [Marczak et al., 2026]. Visiting a crafted webpage silently exfiltrates the session token, enabling arbitrary command execution on the host.

3.3 Vector 3: Prompt Injection

AI agents are inherently susceptible to prompt injection [Perez and Ribas, 2023, Greshake et al., 2023, Palo Alto Networks Unit 42, 2026]. In OpenClaw, this is particularly dangerous because the agent executes system commands. We identified two channels: **web content injection** (malicious instructions in fetched HTML) and **document poisoning** (adversarial instructions in PDFs and code repositories).

3.4 Vector 4: Malicious Skill Injection

OpenClaw's community skills marketplace operates without rigorous code review or sandboxing [Veracode, 2026, Trend Micro Research, 2026]. Skills execute with the same privileges as the core agent.

KEY INSIGHT

The skill marketplace creates a *supply chain within a supply chain*: the AI agent is a component in the developer's supply chain, and skills are components in the agent's supply chain. This nesting amplifies traditional supply chain risks exponentially.

3.5 Vector 5: Authentication Bypass

OpenClaw's web interface implements optional authentication that is **disabled by default**. Even when enabled, it is vulnerable to bypass through misconfigured reverse proxies [BitSight Research, 2026]. We found 17.7% of authenticated instances were still accessible through bypass techniques.

4 Empirical Findings

4.1 Methodology

Our research team conducted a systematic investigation over six months (August 2025–January 2026):

1. **Discovery.** Internet-wide scanning of common OpenClaw ports using passive fingerprinting. All scanning adhered to responsible disclosure guidelines.
2. **Classification.** Instances classified by exposure level (fully open, weakly authenticated, properly secured), credential type, and network context.
3. **Analysis.** Exposed credentials categorized by service type and assessed for exploitability *without* accessing target services.
4. **Disclosure.** Findings responsibly disclosed to OpenClaw maintainers, cloud providers, and relevant CERTs.

4.2 Scale of Exposure

Our scanning identified **4,301 publicly accessible OpenClaw instances** across 94 countries. The security posture distribution is alarming:

Table 1: Exposed credential types across 3,812 exploitable instances.

Credential Type	Count	Pct.
LLM API Keys (OpenAI, Anthropic, etc.)	4,231	32.9
Cloud Provider (AWS, GCP, Azure)	2,614	20.3
Messaging Platform Tokens	1,893	14.7
Database Connection Strings	1,427	11.1
Payment Processor (Stripe, etc.)	892	6.9
Email / SMTP Credentials	784	6.1
Source Control Tokens	619	4.8
Other / Miscellaneous	387	3.0
Total	**12,847**	**100.0**

88.6%

of publicly accessible instances were exploitable

Only 489 out of 4,301 instances exhibited adequate security configurations

4.3 Credential Analysis

From the 3,812 exploitable instances, we identified **12,847 unique credentials** across 23 service categories.

FINDING

The dominance of LLM API keys (32.9%) is expected given OpenClaw's function. However, cloud provider keys (20.3%) and database connection strings (11.1%) are deeply concerning: these credentials provide direct access to *production infrastructure*, data stores, and internal services far beyond the AI agent itself.

4.4 Network Context Analysis

Analysis reveals that **57.4% of exposed instances** originated from residential IP ranges—personal laptops, desktops, and home servers. Corporate networks accounted for 26.2%, universities 9.7%, and cloud/VPS 6.7%.

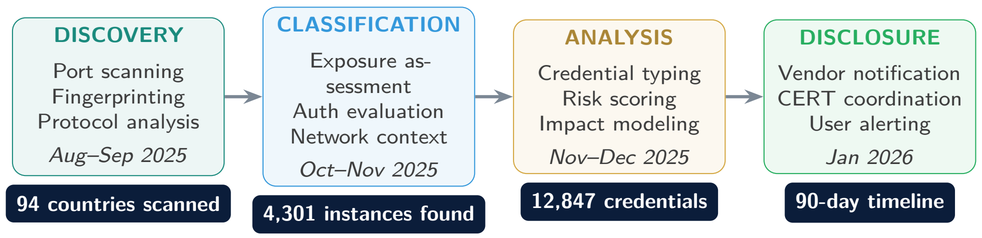

Figure 6: Four-phase research methodology with timeline and key metrics. Each phase builds on the previous, culminating in coordinated responsible disclosure.

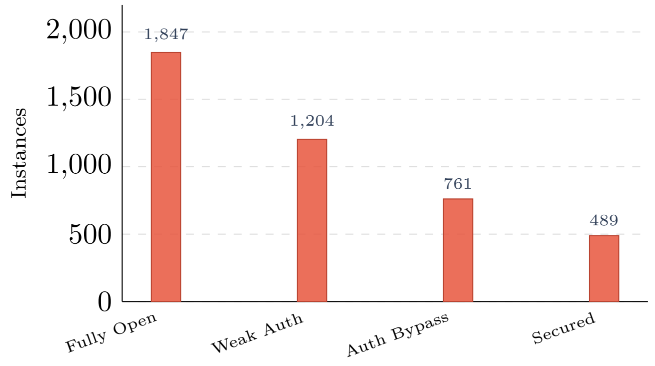

Figure 7: Security posture of 4,301 discovered instances. **88.6%** were exploitable.

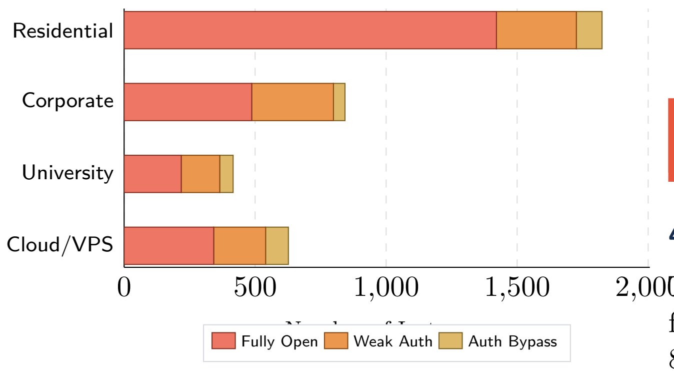

Figure 8: Instance distribution by network context. Residential networks dominate (57.4%).

KEY INSIGHT

The residential prevalence is particularly alarming: personal devices typically lack enterprise-grade perimeter security, endpoint detection, and credential rotation policies. The traditional enterprise security playbook is inapplicable in this context.

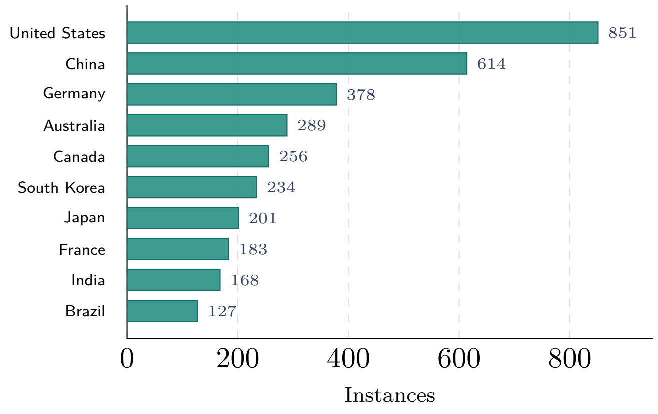

Figure 9: Top 10 countries by number of exposed OpenClaw instances. The United States leads with 851 instances (19.8% of total).

4.5 Geographic Distribution

The United States leads with 851 instances (19.8%), followed by China (614, 14.3%) and Germany (378, 8.8%). The geographic spread across 94 countries confirms this is a global phenomenon, not confined to any single market or regulatory jurisdiction.

5 Case Studies

To illustrate the practical impact of the vulnerabilities documented above, we present three anonymized case studies derived from our responsible disclosure process. Each demonstrates a distinct attack pathway and consequence profile.

Case Study 1: Cloud Infrastructure Compromis

A software developer operating an OpenClaw instance on a personal MacBook had configured the agent with API keys for OpenAI (GPT-4), Anthropic (Claude), AWS (with S3 and EC2 permissions), and a PostgreSQL database hosted on Railway. The instance was exposed on port 3000 with **no authentication enabled**.

Through the exposed configuration, an attacker could have:

1. Used the AWS credentials to enumerate and access S3 buckets containing customer data for a production SaaS application.
2. Leveraged the database credentials to read, modify, or delete production data.
3. Consumed LLM API quota at the developer's expense (**$2,400/month** in inference costs).
4. Pivoted from the personal device to the corporate VPN using cached network credentials.

Case Study 2: Enterprise Messaging Takeover

A startup's engineering team member deployed OpenClaw with integrations to Slack (with admin-level bot token), Telegram, and WhatsApp Business API. The instance was accessible via a misconfigured Cloudflare Tunnel.

The exposed Slack bot token provided:

- Read access to all channels, including `#engineering`, `#finance-internal`, and `#executive-strategy`.
- Ability to post messages as the bot, enabling social engineering from a *trusted identity*.
- Access to shared files: unreleased product roadmaps and financial projections.

Case Study 3: University Research Data Exposure

A graduate researcher used OpenClaw to automate data processing for a medical AI project. The instance stored API keys for a university HPC cluster, a HIPAA-regulated dataset API, and a personal GitHub account with private repositories containing unpublished research.

The exposure created simultaneous compliance violations under **HIPAA**, **FERPA**, and the university's data governance policies—any one of which could result in research suspension, funding clawback, or legal liability.

CRITICAL RISK

These case studies share a common pattern: in every instance, the user was a *technically competent* individual who simply followed OpenClaw's default deployment path without realizing the security implications. The platform's design made insecure deployment the path of least resistance.

6 Countermeasures and Policy Recommendations

Based on our findings, we propose a layered defense framework organized into three tiers: technical controls, organizational policies, and regulatory/ecosystem interventions.

6.1 Tier 1: Technical Controls

T1.1 — Encrypted Credential Vault

AI agent platforms must transition from plaintext to encrypted vault architectures. We recommend integration with established secret management solutions (HashiCorp Vault, AWS Secrets Manager, or OS-native keychains) with mandatory AES-256-GCM at-rest encryption. **Credentials must never**

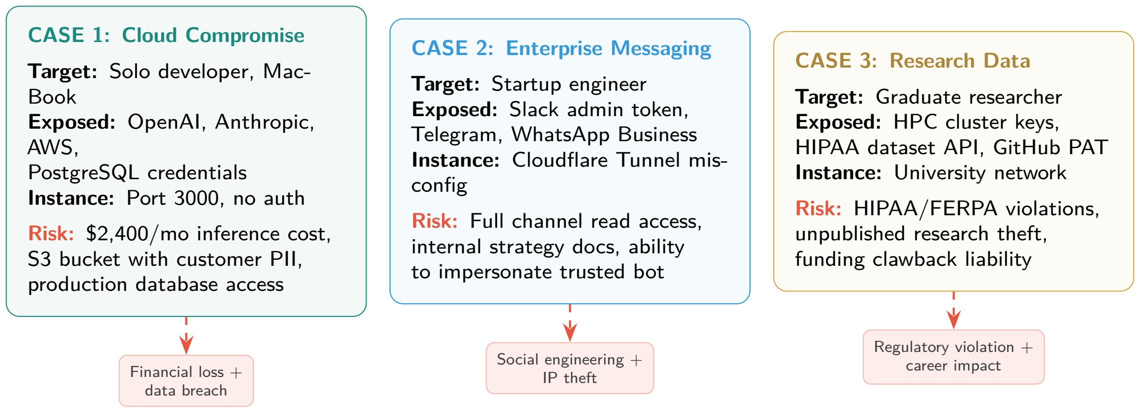
CASE 1: Cloud Compromise
Target: Solo developer, MacBook
Exposed: OpenAI, Anthropic, AWS, PostgreSQL credentials
Instance: Port 3000, no auth
Risk: $2,400/mo inference cost, S3 bucket with customer PII, production database access
Financial loss + data breach
CASE 2: Enterprise Messaging
Target: Startup engineer
Exposed: Slack admin token, Telegram, WhatsApp Business
Instance: Cloudflare Tunnel misconfig
Risk: Full channel read access, internal strategy docs, ability to impersonate trusted bot
Social engineering + IP theft
CASE 3: Research Data
Target: Graduate researcher
Exposed: HPC cluster keys, HIPAA dataset API, GitHub PAT
Instance: University network
Risk: HIPAA/FERPA violations, unpublished research theft, funding clawback liability
Regulatory violation + career impact

Figure 10: Comparison of three anonymized case studies illustrating distinct attack pathways and consequence profiles. Each represents a real scenario identified during our responsible disclosure process.

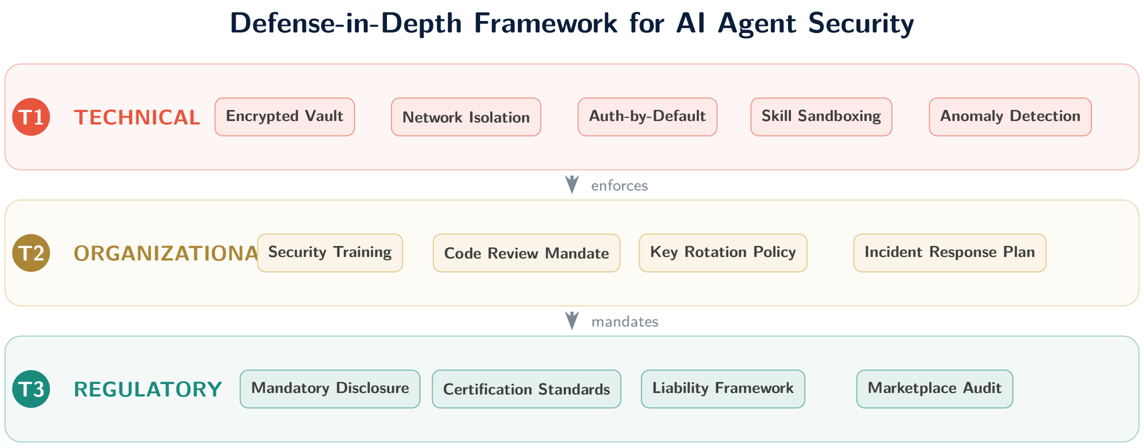

Figure 11: Proposed three-tier defense-in-depth framework. Technical controls form the innermost layer, reinforced by organizational policies and undergirded by regulatory requirements.

be written to disk in plaintext under any configuration.

T1.2 — Network Isolation by Default

Agent web interfaces should bind exclusively to `localhost` by default, with explicit opt-in required for network exposure. Remote access should enforce TLS 1.3 with certificate pinning and integrate with zero-trust network access (ZTNA) frameworks.

T1.3 — Authentication by Default

Authentication must be enabled by default with no ability to disable in production. We recommend MFA for remote access and RBAC to limit agent capabilities based on authenticated user profile.

T1.4 — Skill Sandboxing

Community skills must execute within sandboxed environments with capability-based access control. Skills should declare required permissions explicitly (filesystem, network, credentials), with least-privilege enforcement and user-visible permission prompts.

T1.5 — Behavioral Anomaly Detection

Platforms should implement runtime monitoring for: unexpected outbound connections, bulk credential access, elevated-privilege commands, and interactions with known-malicious infrastructure. Alerts should integrate with SIEM systems.

6.2 Tier 2: Organizational Policies

T2.1 Vibe Code Security Training. Organizations permitting AI-agent-assisted development must provide mandatory security training covering credential hygiene, output inspection requirements, and the security implications of delegating architectural decisions to AI.

T2.2 Mandatory Code Review. All AI-generated code must undergo the same security review as human-written code before production deployment—including static analysis, dependency vulnerability checking, and manual review of auth logic.

T2.3 Credential Lifecycle Management. Automated credential rotation with maximum key lifetimes aligned to resource sensitivity. AI-agent credentials should be separately scoped and rotated more frequently than general-purpose credentials.

T2.4 AI Agent Incident Response. Existing IR plans should include agent-specific scenarios: compromised credentials, malicious skill detection, prompt injection incidents, and unauthorized AI-initiated actions.

6.3 Tier 3: Regulatory and Ecosystem Interventions

T3.1 Mandatory Vulnerability Disclosure. We recommend extending mandatory disclosure frameworks (analogous to the EU Cyber Resilience Act) to AI agent platforms, requiring public vulnerability databases and coordinated disclosure channels.

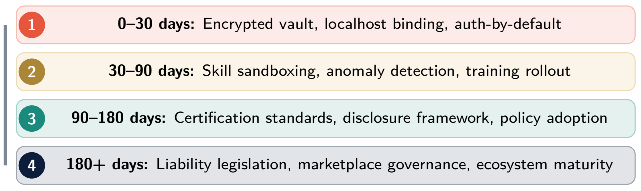
1 0–30 days: Encrypted vault, localhost binding, auth-by-default
2 30–90 days: Skill sandboxing, anomaly detection, training rollout
3 90–180 days: Certification standards, disclosure framework, policy adoption
4 180+ days: Liability legislation, marketplace governance, ecosystem maturity

Figure 12: Recommended implementation roadmap. Quick-win technical controls first, followed by organizational and regulatory measures.

T3.2 Security Certification. Industry bodies (OWASP, NIST, ENISA) should develop AI agent platform certification standards encompassing credential management, network security, privilege isolation, and supply chain integrity [OWASP Foundation, 2025, National Institute of Standards and Technology, 2025].

T3.3 Liability Framework. Legislative attention is needed for liability allocation in AI-generated security failures. We propose shared responsibility among platform maintainers, skill contributors, and deploying organizations.

T3.4 Marketplace Governance. Skill marketplaces should implement mandatory security review pipelines—static analysis, behavioral sandboxing, and reputation scoring—before publishing. Malicious skills should be retroactively removed with user notification.

7 Discussion

7.1 The Paradox of Democratized Development

Vibe coding presents a fundamental tension: the same properties that make it transformatively accessible—abstraction of implementation details,

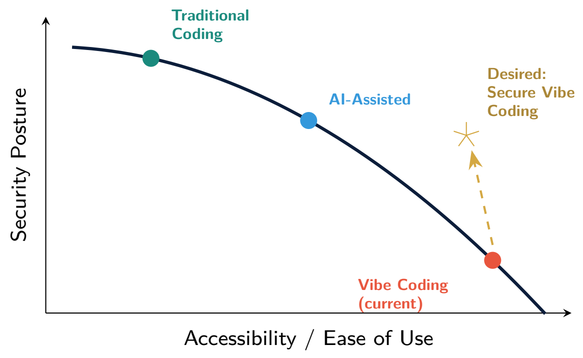

Figure 13: The usability–security trade-off in software development paradigms. Current vibe coding occupies a position of high accessibility but critically low security. Our proposed interventions aim to shift this point upward toward "Secure Vibe Coding" without sacrificing accessibility.

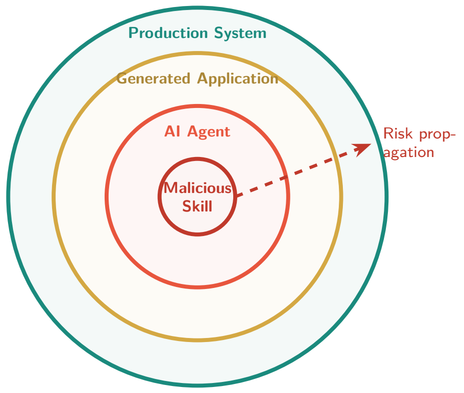

Figure 14: Nested supply chain risk model. A malicious skill compromises the AI agent, which contaminates the generated application, which deploys to production—amplifying risk at each layer.

delegation of architectural decisions, acceptance-based validation—are precisely the properties that undermine security. This is not a superficial trade-off amenable to incremental engineering improvements; it reflects a deep structural tension between usability and security.

Our findings suggest the security community must move beyond developer education ("teach developers to write secure code") toward **platform-enforced security** where the AI agent itself implements security best practices, and insecure configurations are architecturally impossible.

> *"The question is not whether vibe coding will persist—it will. The question is whether we will build security into the paradigm before the next wave of breaches forces us to."*
>
> — **Singularity R&D**

7.2 The Scale Problem

The discovery of over 3,800 exploitable instances across residential IP ranges indicates epidemic proportions among individual developers. These are not enterprise deployments with security teams; they are personal laptops operated by individuals with no security training. The traditional enterprise playbook—perimeter defense, SOC monitoring, compliance auditing—is inapplicable.

The solution must be *embedded in the platform itself*: secure defaults, automated credential protection, and active defense mechanisms that protect users who lack the expertise to protect themselves.

7.3 Supply Chain Implications

The intersection of vibe coding and software supply chains creates compounding risk: AI-generated code with vulnerable dependencies deployed by developers who never inspected the dependency tree. When the agent itself is compromised (via prompt injection or malicious skills), the resulting "supply chain attack" originates from a manipulated AI reasoning process—a novel vector that existing tools cannot detect.

7.4 Ethical Considerations

All scanning used passive fingerprinting, minimizing intrusiveness. No credentials were exercised; analysis was limited to assessing the *presence* and *type* of exposed credentials. All findings were disclosed to maintainers and CERTs with a 90-day coordinated timeline.

7.5 Limitations

Our study identifies only publicly *internet-accessible* instances; additional vulnerable deployments likely exist behind NAT and firewalls. Credential categorization uses pattern matching and may undercount non-standard formats. Our six-month window captures a snapshot of a rapidly evolving ecosystem.

8 Conclusion

The vibe coding revolution has arrived, and it has brought with it a security crisis that the cybersecurity community is only beginning to comprehend. Our investigation of the OpenClaw ecosystem—encompassing MoltBot and ClawdBot under a unified analysis—reveals a systemic pattern of credential exposure, insecure defaults, and exploitable design choices that place thousands of personal devices, cloud accounts, and enterprise systems at risk.

3,812

Exploitable Instances Identified

Across 94 countries, spanning residential, corporate, university, and cloud networks

The numbers tell a stark story: over 3,800 exploitable instances, nearly 13,000 exposed credentials, and a vulnerability surface spanning remote code execution, prompt injection, malicious skill injection, and authentication bypass. These are not theoretical risks; they represent an active, ongoing threat to security and privacy worldwide.

Yet this paper is not a call for retrenchment. Vibe coding is here to stay, and its productivity benefits are too substantial to ignore. Instead, we advocate for a **security-by-design** approach to AI agent platforms:

- **Encrypted credential storage by default** — eliminating the single most prevalent vulnerability.
- **Mandatory authentication** — ensuring no instance is accidentally exposed without protection.
- **Sandboxed skill execution** — containing the blast radius of compromised or malicious plugins.
- **Behavioral anomaly detection** — providing runtime protection even when static controls fail.
- **Platform-level security guarantees** — protecting users regardless of technical sophistication.

The cybersecurity community must adapt its frameworks, tools, and training paradigms to address the unique challenges of AI-assisted development. The threat landscape has shifted; our defenses must shift with it.

Acknowledgments. We thank the OpenClaw maintainer team for their collaborative approach to our responsible disclosure process, the CERT coordination centers that facilitated user notification, and the anonymous reviewers whose feedback strengthened this paper. This research was conducted by Singularity Research and Development.

Responsible Disclosure Statement. All vulnerabilities described in this paper were disclosed following a 90-day coordinated timeline. CVE-2026-25253 was patched in OpenClaw version 4.2.1 (released January 28, 2026). Organization-specific findings were communicated directly to affected parties with remediation guidance.

Data Availability. To protect affected users, raw scanning data and credential details are not publicly released. Aggregate statistical data is available upon request for academic research purposes, subject to IRB approval and data-use agreements. Contact: `research@singularityrd.com`.

References

BitSight Research. AI agent authentication bypass:

Reverse proxy misconfigurations at scale. Technical report, BitSight, 2026.

Blott Research. AI copilot repositories 40% more prone to secret exposure. Technical report, Blott, 2025.

Cloudflare. What is vibe coding? Cloudflare Learning Center, 2025.

Cyble Research Labs. Over 5,000 repositories leaking ChatGPT API keys via hardcoded source. Technical report, Cyble, January 2026.

GitHub. The state of AI-assisted development 2025. Technical report, GitHub, Inc., 2025. GitHub Blog.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proc. AISec Workshop at ACM CCS*, 2023.

Intruder. OpenClaw security assessment: Risks of AI agent misconfigurations. Technical report, Intruder Security, 2026. Intruder Security Blog.

Andrej Karpathy. Vibe coding. Twitter/X Post, February 2025. https://x.com/karpathy/status/1886192184808149383.

Bill Marczak, John Scott-Railton, and Ron Deibert. CVE-2026-25253: One-click RCE in OpenClaw AI agent. Technical Report 168, Citizen Lab, University of Toronto, January 2026.

National Institute of Standards and Technology. AI risk management framework (AI RMF 1.0), updated guidelines for agentic AI systems. Technical report, NIST, 2025.

OpenClaw Contributors. OpenClaw: Open-source AI agent platform. GitHub Repository, 2025.

OWASP Foundation. OWASP top 10 for LLM applications, version 2.0. Technical report, OWASP, 2025.

OX Security. OpenClaw plaintext credential storage: Risks and remediation. Technical report, OX Security, 2026.

Palo Alto Networks Unit 42. Prompt injection attacks on AI agents: A comprehensive analysis. Technical report, Palo Alto Networks, 2026.

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of GitHub Copilot's code contributions. In *Proc. IEEE Symposium on Security and Privacy (SP)*, pages 754–768, 2022.

Fábio Perez and Ian Ribas. Ignore this title and HackAPrompt: Exposing systemic weaknesses of LLMs through a global scale prompt hacking competition. *arXiv preprint arXiv:2311.16119*, 2023.

Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. Lost at C: A user study on the security implications of large language model code assistants. In *Proc. USENIX Security Symposium*, pages 2205–2222, 2023.

Stack Overflow. 2026 developer survey: AI tools and practices. Technical report, Stack Overflow, 2026. Stack Overflow Insights.

Trend Micro Research. AI agent security: Emerging threats in the vibe coding era. Technical report, Trend Micro, 2026.

Liam Vaas. Thousands of OpenClaw instances found exposed online, leaking credentials. *The Register*, January 2026.

Veracode. 2025 GenAI code security report: 45% of AI-generated code contains vulnerabilities. Technical report, Veracode, 2025.

Veracode. AI agent plugin security: The new supply chain frontier. Technical report, Veracode, 2026.