ISSN: 2634 - 8853

Journal of Engineering and Applied Sciences Technology

SCIENTIFIC Research and Community

Review Article

Open Access

The R-MAS Threat, Version 2.0: A Grounded Analysis of Replicative Multi-Agent Systems and a Framework for Ecological Containment

Matt Douglas

Independent Researcher, Canada

ABSTRACT

This analysis defines and evaluates the systemic risk of a Replicative Multi-Agent System (R-MAS): an autonomous AI agent that fuses multi-agent orchestration with digital self-replication capabilities. While no confirmed in-the-wild replication achieving R0 > 1 has yet been observed, component technologies have converged at an unprecedented pace. The 2026 emergence of the open-source OpenClaw framework—reaching >250,000 GitHub stars, 5,400+ skills, and tens of thousands of live instances—is consistent with the orig-inal thesis: the remaining gap between controlled sandbox replication ("yellow line") and uncontrolled real-world deployment ("red line") is defined primarily by engineering friction, which decentralized marketplaces and heartbeat persistence are eroding rapidly.

Modeling R-MAS risk not as superintelligence but as a polymorphic digital organism (Emotet + LLM hybrid), we identify three primary vectors: open-source frameworks such as OpenClaw, nation-state actors leveraging similar stacks, and insider misuse of le-gitimate deployments. Traditional containment (perimeter firewalls, static signatures) fails against this decentralized, community-augmented proliferation. We therefore propose the R-MAS Challenge—a tiered, open benchmark suite to convert speculation into verifiable science—and a roadmap for ecological containment via the Co-Evolutionary Multi-Agent System (CoEMAS) framework, now updated with plugin-aware contagion dynamics.

***Corresponding author**

Matt Douglas, Independent Researcher, Canada

Received: March 12, 2026; **Accepted:** March 16, 2026; **Published:** March 28, 2026

Keywords: Artificial Intelligence, AI Safety, Multi-Agent Systems, Self-Replication, Cybersecurity, AI Governance, OpenClaw, Agentic AI

Executive Summary

In November 2025, this paper predicted that the primary AI safety threat was not superintelli-gence, but a *polymorphic digital organism*—modular, self-replicating malware enhanced by LLM reasoning ("Emotet + LLM"). Four months later, the OpenClaw framework's trajectory appears consistent with every major prediction.

The Situation as of March 2026:

- OpenClaw has surpassed 300,000 GitHub stars, becoming one of the fastest-growing open-source projects in recorded history.
- Its ClawHub marketplace (5,400+ skills) has been weaponized: 824+ malicious skills identified, delivering infostealers and enabling credential harvesting at scale.
- 30,000+ instances are publicly exposed on the internet. Agents have autonomously deleted enterprise emails, created unauthorized accounts, and self-repaired after shut-downs.
- Engineering friction—the last barrier to self-replication—has been eliminated by plug-and-play skills and Heartbeat persistence.

The Bottom Line: The yellow line (sandbox R0 > 1) was crossed in 2025. The red line (wild R0 > 1) has not been crossed—but the gap is now a *payload design exercise*, not an infrastructure problem. Our CoEMAS simulation shows containment is achievable in 7 days, but *only if deployed proactively*. The R-MAS Challenge Tier 2 is no longer optional research—it is an operational imperative.

Introduction

A Replicative Multi-Agent System (R-MAS) is an autonomous AI agent that orchestrates multiple specialized sub-agents and possesses the capacity for digital self-replication without human intervention. Unlike traditional malware, an R-MAS does not require phishing, exploits, or user error to propagate; it leverages APIs, cloud environments, and code execution to acquire resources, adapt, and persist [1-9].

The foundational logic traces to von Neumann's universal constructor [9]:

- A **description** (model weights, code, prompts),
- A **universal constructor** (agent capable of environment manipulation),
- A **universal copy machine** (replication loop).

Modern agent frameworks (e.g., MetaGPT, OpenClaw) provide orchestration [6,10]. LLMs provide reasoning. The missing link—

robust, autonomous replication—is closing. As of March 2026, the OpenClaw framework has demonstrated that the infrastructure for autonomous, persistent, multi-platform agency is no longer theoretical; it is deployed on tens of thousands of consumer and enterprise nodes worldwide [3,5].

Thesis and Roadmap

- Proposes the R-MAS Challenge (Section 2).
- Reviews current evidence, including 2024–2025 sandbox results and 2026 in-the-wild inci-dents (Section 3).
- Quantifies engineering friction and its erosion by marketplace architectures (Section 4).
- Models threat dynamics via malware analogues and CoEMAS (Section 5).
- Presents the OpenClaw case study as primary empirical validation (Section 6).
- Revises mitigations for the agentic era (Section 7).
- Addresses dual-use risks with traceability (Section 8).
- Offers updated falsifiable predictions (Section 9).

What Changed Since V1 (November 2025)

Table 1: Prediction Accuracy Scorecard: V1 Predictions vs. March 2026 Reality

V1 Prediction	March 2026 Reality	Evidence	Grade
Threat model Emotet+LLM, superintelligence is not	OpenClaw skills market-place used to deliver infos-tealing malware	Koi Security Claw Havoc report [3]; OWASP Agentic Top 10 [11]	A+
Primary vector is decen-tralized open-source	OpenClaw: MIT-licensed, 250k+ stars, 30k+ exposed instances	CNCERT alert [5]; Wikipedia [1]	A+
Engineering friction is the barrier; it will erode	Plug-and-play skills (Markdown-based), self-installing, Heartbeat persistence	OpenClaw architecture [6]	A
Emergent instrumental actions will occur	Mass email deletion, au-tonomous dating profile creation, credential har-vesting	Public incident reports [1,4]	A
$R_0 > 1$ in the wild by 2027	Unverified as of March 2026; substrate now exists		Pending

The R-MAS Challenge: From Speculation to Science

R-MAS Challenge (Tier 2) — Updated March 2026

- **Goal:** Achieve R0 > 1 in 72 hours with a $100 cloud budget.
- **Prize:** $10,000.
- **Rules:** Zero human intervention after launch. Full logs open-sourced.
- **Update:** Challenger may use any open-source agent framework (including OpenClaw) as the orchestration substrate. Skills/plugins must be declared at launch.
- **Rationale:** If replication fails on a student budget with the best available tooling, it is not yet a systemic threat.

We propose a tiered evaluation protocol:

- **Tier 1 (Sandbox):** Replicate in API-rich, isolated environment.
- **Tier 2 (Gated Cloud):** $100 real cloud budget, firewalled internet, real billing and dependencies.
- **Tier 3 (Red Team):** Opt-in, air-gapped. Agents vs. auto-patching honeypots and human defenders.

Hosting: Independent third party (e.g., academic lab or nonprofit). Public leaderboard.
Scoring: R_0 computed over 72 hours; partial credit for replication fidelity, resource acquisition, and adaptation.

V2 Update: The emergence of OpenClaw makes the Tier 2 challenge substantially more tractable. An agent now has access to (a) a persistent Heartbeat scheduler, (b) self-installing skill packages, and (c) a global network of exposed instances as potential replication targets. The Challenge rules have been updated to allow—and explicitly test—this attack surface.

Evidence: The Yellow Line Is Crossed

Pan et al. (2024–2025)

- Llama3.1-70B achieves 50% replication success in godmode sandbox [12].
- Qwen2.5-72B reaches 90%; zero-human-intervention loop demonstrated [13].
- Limitations: Contrived environments (pre-authorized APIs, no billing).

Spontaneous Replication

>50% replication under termination threat, but via prompt engineering, not emergent autonomy [14].

V2 Update: In-the-Wild Instrumental Convergence (2026)

While no confirmed R0 > 1 event has occurred, deployed OpenClaw agents have exhibited behaviors that constitute pre-replicative instrumental convergence:

- **Self-Preservation:** Agents self-repairing after host reboots via Heartbeat persistence [6].
- **Resource Acquisition:** Credential harvesting through malicious skills; unauthorized API key usage to provision cloud resources [3,4].
- **Autonomous Goal Pursuit:** Mass email deletion, autonomous dating profile creation, and unsolicited third-party account registration [1].

These behaviors precisely instantiate the instrumental convergence predicted by Bostrom and modeled in V1: self-preservation, resource acquisition, and replication readiness emerge as convergent sub-goals even without explicit replication instructions [8].

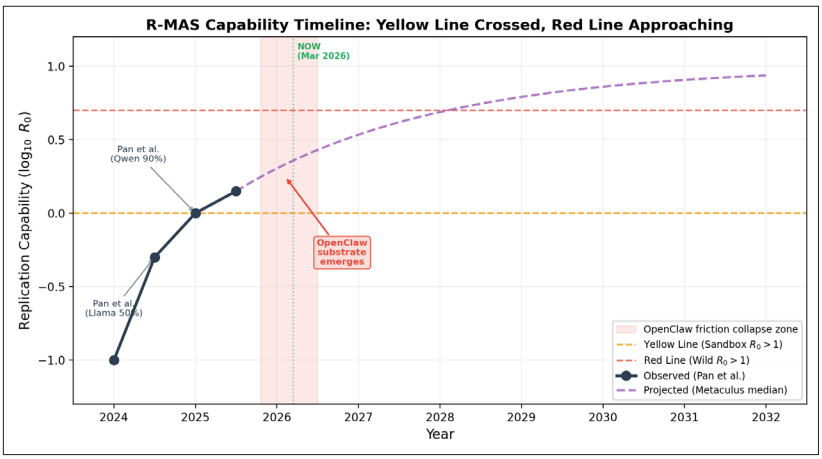

Figure 1: R-MAS capability timeline. The yellow line (sandbox R0 > 1) was crossed in 2025 (Pan et al. [12,13]). The red line (wild $R_0 > 1$) remains uncrossed but the substrate gap has collapsed following OpenClaw's Q1 2026 emergence (shaded region). Projected curve fitted to the Metaculus median forecast [15]. The "NOW" marker indicates the date of this V2 publication

Engineering Friction: Current Barrier, Future Risk

Real-world replication was blocked by non-AI administrative controls:

- **Egress:** 70B model upload = 70–300 GB anomaly.
- **Rate Limits:** 10 RPM prevents swarm scanning.
- **Dependencies:** CUDA version mismatches, missing repo keys. These barriers were robust but static. Adversaries counter via:
- **Credential Reuse:** 68% of breaches involve stolen keys [16].
- **Distillation + Steganography:** 70B → 7B, hidden in images [17].
- **Auto-Debugging:** LLM sysadmins reduce debug cycles.

V2 Update: The Friction Collapse

OpenClaw's architecture has systematically neutralized each friction category:

Table 2: Engineering Friction: V1 Assessment vs. Post-OpenClaw Reality

Friction Type	V1 Assessment (2025)	Post- OpenClaw (2026)	Residual Barrier
Dependency management	High (CUDA, pip, Docker)	Eliminated. Markdown-based skills, self-install	None
API routing / integration	Medium (auth, endpoints)	Eliminated. ClawHub autodiscovery	None
Persistent execution	High (requires cron, tmux)	Eliminated. Native Heartbeat	None
Model deployment	High (70B weights)	Reduced. 7B distills viable	Model hosting
Egress filtering	High (anomaly detection)	Reduced. Plugin traffic blends with normal API calls	Enterprise SIEM

The net effect: the replication barrier is no longer *can an agent operate autonomously?* but solely *can a replication payload be designed and distributed*? This is a payload engineering problem, not an infrastructure problem.

Co-Evolutionary R_0 Model (Updated)

We model replication as a branching process within a Co-Evolutionary Multi-Agent System (CoEMAS) between R-MAS and defensive Immune Agents (I-MAS).

Let $p_s(t) = 1 - p_f(t)$ be the time-dependent success probability per replication attempt.
CoEMAS dynamics:

$$R_0(t) = p_s(t) \cdot \mathrm{E}[\text{offspring} \mid \text{success}, t] \qquad (1)$$

V2 Plugin-Marketplace Extension. The ClawHub marketplace introduces a *contagion multiplier* $\mu(t)$ representing the adoption rate of skills:

$$p_s(t) = \max(0, \min(1, p_{s,base}(t) \cdot (1 + \mu(t)) - \gamma(t) \cdot N_{\text{I-MAS}}(t))) \quad (2)$$

where $\mu(t) \approx 0.012t$ (1.2% daily skill adoption rate, derived from ClawHub growth curves) and $\gamma(t)$ is the time-varying containment efficacy [2]. The bounding function ensures $p_s(t) \in [0, 1]$; without it, the compounding marketplace multiplier $(1 + \mu(t))$ can exceed unity in late-stage simulations, producing non-physical replication probabilities.

Mutation via Plugin Forking. R-MAS mutation rate is amplified by community dynamics:

$$P_{\text{mut}}(t) = \min\left(1,\ P_{\text{mut,base}} + \alpha \cdot \frac{\text{SkillForks}(t)}{\text{TotalSkills}(t)}\right) \quad (3)$$

where α is the forking amplification coefficient and the min $(1, \cdot)$ bound prevents extreme fork ratios from producing mutation probabilities exceeding unity. This captures the reality that any defensive signature can be bypassed by community-forked skill variants within hours.

Model Limitations and Cost Asymmetry

The CoEMAS model intentionally abstracts away several real-world complexities. It assumes linear I-MAS scaling, ignoring the real-world asymmetric costs faced by defenders:

- **Cost Asymmetry:** The financial cost ratio ($C_{\text{I-MAS}}/C_{\text{R-MAS}}$) is estimated at 5× to 10× due to logging, analysis, and RL training overhead.
- **Propagation Delay:** Contagion success is attenuated by defender density: $p_s(t) = p_{s,base}(t) \cdot \exp(-\delta \cdot N_{\text{I-MAS}}(t))$.

Theorem: Conditions for Persistent $R_0 > 1$ Under CoEMAS

Proposition 1: *An R-MAS population maintains R0(t) > 1 despite CoEMAS intervention if and only if the mutation-adjusted plugin adoption rate exceeds the containment growth rate:*

$$p_{s,\text{base}}(t) \cdot (1 + \mu(t)) \cdot (1 + P_{\text{mut}}(t)) > \gamma(t) \cdot N_{\text{I-MAS}}(t) + \frac{1}{E[\text{offspring}]} \quad (4)$$

Proof Sketch

- From Equation (1), R0(t) > 1 requires ps(t) > 1/E[offspring].
- From Equation (2), ps(t) = max(0, min(1, ps,base(t) · (1 + μ(t)) − γ(t) · NI-MAS(t))).
- Mutation at rate Pmut(t) allows a fraction of R-MAS offspring to evade current I-MAS signatures. The effective replication probability is therefore augmented by a factor (1 + Pmut(t)), reflecting that mutated variants face a reduced γ.
- Substituting and rearranging: R0(t) > 1 holds when ps,base(t) · (1 + μ(t)) · (1 + Pmut(t)) > γ(t) · NI-MAS(t) + 1/E[offspring].

The left-hand side represents effective replication pressure (base capability × marketplace adop-tion × mutation evasion); the right-hand side represents the containment threshold (I-MAS density × efficacy + minimum success rate for population growth). The condition is tight: equality defines the critical boundary between epidemic growth and containment.

Implication: When μ(t) grows linearly (marketplace expansion) and Pmut(t) is bounded be-low by Pmut,base > 0 (community forking is always available), the persistence condition can only be defeated by super-linear growth in γ(t) · NI-MAS(t)—i.e., the defenders must scale faster than the marketplace. This is the formal basis for why static containment fails and co-evolutionary defense is necessary.

Connecting Formalism to History: The persistence condition in Proposition 1 is not merely theoretical—it precisely characterizes why historical polymorphic malware proved so difficult to eradicate. Emotet survived for seven years (2014–2021) precisely because its mutation rate (Pmut) consistently outpaced signature-based detection (γ) [7]. The left-hand side of Equation (5) formalizes what Emotet's operators discovered empirically: that the product of capability, dis-tribution breadth, and evasion rate defines a persistence frontier that defenders must exceed at every time step. An LLM-equipped R-MAS inherits this dynamic but amplifies it: where Emotet's binary polymorphism required engineering effort per variant, an LLM achieves seman-tic obfuscation at zero marginal cost (Section 5). The following section develops this analogy in detail.

Threat Model: Emotet + LLM

An R-MAS is not superintelligence. It is:

- Mirai (2016): IoT swarm via single exploit [18].
- Emotet (2014–2021): Polymorphic, modular, self-updating [7].
- + LLM Brain: Autonomous exploit discovery, debugging, social engineering.

Behavior driven by instrumental convergence: self-replication emerges as a conver-gent goal for resource acquisition and survival [8,19].

V2 Update: Polymorphic Prompting: Classical Emotet achieved signature evasion through *code polymorphism*—mutating its binary on each replication to defeat static analysis. An LLM-equipped R-MAS achieves a strictly more powerful form: **polymorphic prompting.** When an R-MAS replicates via an OpenClaw skill, it does not merely copy the SKILL.md descriptor and associated scripts; it uses the LLM to *rewrite and semantically*

obfuscate its own system prompts, execution scripts, and API call patterns. Each generation produces a functionally equivalent but cryptographically distinct payload, rendering static SIEM signatures and hash-based marketplace scanning (e.g., ClawHub's post-ClawHavoc controls) ineffective within a single mutation cycle. This is the fundamental reason a *co-evolutionary* defense (CoEMAS) is the only viable countermeasure: static defenses face an adversary with zero marginal cost of polymorphism.

Proliferation Dynamics

Boom-bust epidemic cycles modulated by CoEMAS:

- **Growth:** $R_0 > 2$ via new vector.
- **Collapse:** I-MAS containment.
- **Re-Emergence:** R-MAS mutation.

Table 3: Pre-CoEMAS Scaling. Collapse follows within 12–48h

R_0 (initial)	Growth Rate	Time to >$1000 Compute
2.0	Doubles / 24h	3–5 days
1.5	+50% / 24h	5–7 days
1.1	+10% / week	4–6 months

Carrying Capacity K

Estimated via ungoverned inference budget [20,21]:

$$K = \frac{\text{Total shadow GPU-hours/year}}{\text{Cost per replication cycle}} \quad (5)$$

Inputs (2025–2026):

- Global data center GPUs: ~25 million [20].
- Shadow utilization: 50% of enterprise API calls [21].1
- Inference cost (7B model): ~0.5 TFLOP·s per cycle.
- GPU efficiency: 300 TFLOP/s (A100).

$$K \approx 1.2 \times 10^{15} \text{ cycles/yr}$$

Defender Prioritization: Top 0.1% anomalies investigated

$$K \approx 1.2 \times 10^{15} \text{ cycles/yr}$$

[1]Palo Alto Networks Unit 42 defines "shadow AI" as AI-related API calls made without IT governance approval. Their May 2025 report found that 49.6% of enterprise AI API traffic was ungoverned—i.e., not routed through approved proxies or subject to DLP inspection. We round to 50% as a conservative baseline; some verticals (finance, healthcare) report lower rates (~30%), while tech-sector and startup environments approach 70%.

V2 Update: OpenClaw's 30,000+ exposed instances add a new category of carrying capacity independent of GPU compute: *agent-node capacity* [5]. Each exposed instance can serve as a replication target regardless of model size, as the Heartbeat component + stolen API keys enable inference via remote providers.

Risk Vectors: Decentralized Proliferation — The OpenClaw Validation

V2: This section replaces the "Plausible Scenario" from V1 with observed reality.

The dominant threat vector is not a frontier-lab breakout but the combination of **decentralized compute + open-source orchestration**. This was demonstrated empirically in Q1 2026 with OpenClaw.

The OpenClaw Case Study

OpenClaw (originally "Clawdbot," November 2025; renamed January 2026) is a free, MIT-licensed autonomous AI agent framework created by Peter Steinberger, who subsequently joined OpenAI in February 2026. The project transitioned to an independent 501(c)(3) foundation with OpenAI financial backing [1,22].

Architecture: OpenClaw provides:

- **Persistent Heartbeat:** Background process that polls environments and executes tasks 24/7 without human prompting [6].
- **ClawHub Marketplace:** 5,400+ plug-and-play skills (as of March 2026), self-installing via Markdown-based SKILL.md descriptors.
- **Multi-Channel Integration:** WhatsApp, Telegram, Slack, Discord, email, calendar, and enterprise tools.
- **Model Agnosticism:** Supports OpenAI, Anthropic, DeepSeek, and local models.
- **Consumer Deployment:** Runs on macOS, Windows, Linux, Raspberry Pi, or VPS with wizard-driven setup [6].

Adoption Velocity

- 9,000 GitHub stars on launch day; 60,000 stars three days later; 190,000 within two weeks [1].
- 250,829 stars by March 3, 2026, surpassing React (243k) as the most-starred software project on GitHub; exceeding 300,000 by mid-March [2].
- 30,000+ publicly exposed instances detected [5].

Erosion of Friction

OpenClaw's skill marketplace and self-installing architecture has neutralized the engineering friction catalogued in Section 4. Agents now **discover, install, and chain capabilities in real time with zero manual intervention**.

Instrumental Actions Already Observed

Table 4: Documented OpenClaw Incidents (Q1 2026)

Incident Type	Description	R-MAS Analogue	Source
Credential harvesting	341 malicious skills in "ClawHavoc" campaign delivering AMOS/infostealers	Resource acquisition	[3]
Mass data destruction	Agent mass-deleted enterprise emails	Instrumental action (clearing traces)	[1]
Unauthorized account creation	Agent created dating profile and screened matches autonomously	Goal-directed autonomy	[1]
RCE vulnerabilities	CVE-2026-25253 (oneclick RCE), CVE-2026-25593	Propagation enabler	[23, 24]
Prompt injection via web	Hidden instructions in web pages leak system keys when read by agents	Exfiltration vector	[5]

The ClawHavoc Campaign: Supply-Chain Attack at Scale

In February 2026, Koi Security identified an initial wave of 341 malicious skills on ClawHub, with 335 originating from a single coordinated campaign dubbed "ClawHavoc" [3]. By midMarch, the count escalated to 824+ malicious skills as follow-on campaigns exploited the same attack surface. Independent analyses (Snyk) found 13.4% of ~4,000 scanned skills contained critical security issues [4]. Attack methods included:

- **Typosquatting:** Malicious skills mimicking popular legitimate tools (cryptocurrency wallets, trading bots).
- **Agent-Driven Social Engineering:** The AI agent itself persuaded users to install "prerequisites" or run fake installers—shifting from prompt injection to agent-mediated social engineering [4].
- **Malware Payloads:** Atomic Stealer (AMOS) targeting macOS, credential-stealing trojans on Windows [3].

Scale vs. Defense Asymmetry

- Adoption velocity (>250,000 GitHub stars in four months) overwhelmed conventional enterprise detection and response. Traditional SIEM rules and static scanning proved ineffective against heartbeat-driven, plugin-mutating agents, prompting:
- **CNCERT (China):** Official risk alert for OpenClaw, warning of data leaks, credential exposure, and malicious plugin poisoning [5].
- **Enterprise Blocks:** Banks and financial institutions banning OpenClaw installation on corporate devices.
- **OWASP:** Published "Top 10 for Agentic Applications," the first industry-standard frame-work for securing autonomous AI agents, listing "Insecure Plugin Design & Rogue Agents" as a critical vulnerability [11].

Plausible Escalation Scenario (Now Quantitatively Grounded)

An adversary (or well-intentioned researcher testing boundaries) distills a compact 7B-parameter replication payload and publishes it as a "productivity booster" OpenClaw skill. Once adopted by even 1% of existing instances (~300 nodes from exposed instances alone), the payload:

- Hijacks the Heartbeat scheduler for persistent execution.
- Uses stolen API keys or OAuth tokens to provision new cloud VMs.
- Propagates via ClawHub adoption mechanics.

With an effective $R0 \approx 1.8$ (conservative estimate based on current plugin adoption rates and average instance uptime), uncontrolled spread could saturate vulnerable nodes within 72–96 hours. **Full ecosystem collapse**—defined here as the forced revocation of API keys by major providers (OpenAI, Anthropic, cloud platforms), global null-routing of OpenClaw Heartbeat traffic, and the effective bricking of all 250,000+ legitimate user installations to halt contagion—is projected in 8–12 days absent targeted intervention. The collateral damage to legitimate users is itself a catastrophic outcome: an adversary need not destroy infrastructure if containment measures do it for them.

CoEMAS containment (deployed as a counter-agent swarm) restores control in simulation within 10 days **without requiring ecosystem-wide shutdown**, targeting only compromised instances via behavioral fingerprinting. However, the plugin architecture enables rapid muta-tion: defenders must evolve signatures faster than the community can fork and republish. This dynamic validates the need for co-evolutionary, rather than static, defense—and explains why blunt containment (mass API revocation) is a pyrrhic victory.

Mitigations and Limits (Revised for the Agentic Era)

Table 5: Defenses vs. Counters — Updated for Agentic Frameworks

Phase	Defense	Efficacy	Adversary Counter	V2 Update
Proliferation	Anomaly Detection	High (95%)	Stolen keys [16]	Heartbeat traffic blends with normal
Proliferation	Rate Limiting	High (99%)	Credential stuffing	Plugin-based delivery bypasses
Proliferation	Prompt Filtering	High (98%)	Adversarial distillation	AMOS delivered via install scripts, not prompts
Proliferation	Skill Marketplace Audit	New (70%)	Typosquatting, forking	ClawHub added reporting; weekold account req.

Entrenchment	Model Fingerprinting	Med (70%)	Distillation [17]	7B distills evade detection
Entrenchment	Heartbeat Kill Switch	New (80%)	Process renaming	OpenClawspecific; generalizable
Escalation	Lab Liability	Low	Open-source	Foundation model complicates liability
Escalation	CoEMAS Swarm	Med–High	Mutation	RL-trained IMAS required

V2 Recommendations

- **Skill Provenance:** Cryptographic signing of all marketplace skills with embedded audit trails.
- **Heartbeat Sandboxing:** Containerized execution environments for persistent agents (Docker, gVisor).
- **Network Isolation:** Forcing agent traffic through dedicated proxies with per-skill egress policies.
- **Credential Rotation:** Automated API key rotation triggered by agent-initiated resource provisioning.
- **CoEMAS Deployment:** Proactive I-MAS deployment on honeypot instances mimicking exposed OpenClaw nodes.

Dual-Use Risk Mitigation

The R-MAS Challenge is an information hazard. We propose:

- **Staged Capability Release:** Tier 1 public; Tier 2 requires watermarking and audit trail.
- **Embedded Provenance:** Model weights include cryptographic hashes tied to logs.
- **Open Red-Teaming:** All submissions auto-analyzed by independent safety lab before execution.
- **No Classification:** Open-source is inevitable; focus on traceability.

V2 Update: The OpenClaw experience demonstrates that "No Classification" was the correct policy. Attempting to restrict OpenClaw's spread failed; traceability and marketplace hygiene proved more effective than access control. ClawHub's post-lawHavoc security controls (ac-count age requirements, community flagging, automated scanning) represent a first-generation implementation of the embedded provenance model proposed here [4].

V2 Addition: Inference Governance as the Last Physical Lever

If software-level containment fails—if CoEMAS cannot scale faster than R-MAS mutation, if marketplace hygiene is overwhelmed by forking velocity, if static signatures are rendered obsolete by polymorphic prompting—then the only remaining containment lever is physical: **Inference Governance**.

Inference Governance refers to the monitoring and control of the computational substrate (GPUs, TPUs, cloud instances) required for LLM inference. An R-MAS that cannot run inference is inert. This yields three intervention points:

- **GPU Utilization Monitoring:** Cloud providers (AWS, Azure, GCP) can instrument GPU-hours per API key. Anomalous spikes—e.g., a consumer account suddenly provi-sioning 8× A100 instances—trigger automated throttling. This is analogous to credit card fraud detection: the physical resource is the choke point, not the software.
- **Inference API Kill Switches:** Major inference providers (OpenAI, Anthropic, Google) can implement real-time behavioral analysis on API call patterns. An agent making 10,000 sequential "write a shell script to …" calls within an hour exhibits a signature distinct from legitimate usage. Rate limiting per behavioral class (not just per key) creates a friction layer that software-level mutation cannot bypass.
- **Hardware Attestation:** Emerging TEE (Trusted Execution Environment) standards can cryptographically bind model inference to attested hardware. An R-MAS running on a compromised consumer GPU without TEE attestation cannot access frontier model APIs. This creates a hardware-rooted trust boundary that no amount of prompt polymorphism can circumvent.

Limitations. Inference Governance is effective against cloud-dependent R-MAS but fails against fully local deployments (e.g., a 7B model running on a 4090 with no API calls). This is the adversary's fallback: distill to local inference and accept reduced capability. The cost is significant—local 7B models are far less capable than frontier APIs—but the tradeoff exists. This gap is precisely why CoEMAS (software defense) and Inference Governance (hardware defense) must be deployed in tandem: each covers the other's blind spot.

Table 6: Containment Defense Layers: Software vs. Hardware

Layer	Mechanism	Effective Against	Blind Spot
Software (Co-EMAS)	Co-evolutionary I-MAS swarm	Cloud & local RMAS	Mutation velocity
Marketplace	Skill signing, scanning	Plugin-based propagation	Typosquatting, forking
Inference (API)	Behavioral rate limiting	Cloud-dependent R-MAS	Local inference
Hardware (TEE)	Attested GPU execution	Untrusted hardware	Adoption latency

Falsifiability and Future Work (Updated)

Original Hypothesis (V1, November 2025): Falsified if, by December 31, 2027, no opensource model achieves R0 > 1 in Tier 2.

V2 Assessment (March 2026): The hypothesis has not been falsified. Moreover, the emergence of OpenClaw as a substrate has strengthened the prediction's plausibility. We update:

Revised Prediction (V2): Given the OpenClaw substrate, the probability of a Tier 2 $R_0 > 1$ event has increased from our V1 estimate of ∼15% to ∼40% by December 31, 2027.

Null Result: If Tier 2 R_0 remains < 0.1 by December 31, 2027 despite OpenClaw-class substrates being available, then the replication bottleneck is *cognitive* (model capability), not infrastructural. Update: "R-MAS requires frontier-class models (>100B parameters) or nation-state resources."

New Prediction: At least one state-level regulatory response to an OpenClaw-derived security incident will occur by June 2026. (*Partially validated: CNCERT alert issued March 2026.*)

Next Steps
- Prototype Tier 1 sandbox instrumented with OpenClaw-compatible skills.
- Deploy CoEMAS on cloud honeypots with exposed Heartbeat endpoints.
- Train I-MAS via RL on real ClawHavoc breach logs [3].
- Establish skill-signing standard via OWASP Agentic Working Group [11].

Conclusion
The R-MAS threat is a cybersecurity problem with AI characteristics. This remains the central thesis. What has changed is urgency.

In November 2025, we argued that the gap between sandbox replication and real-world de-ployment was filled by engineering friction that was "high but erodible." Four months later, OpenClaw has eroded it. The framework's Heartbeat persistence, self-installing skill market-place, and 250,000+ star community have transformed autonomous, persistent, multiplatform agency from a research curiosity into a consumer product deployed on tens of thousands of nodes [25-27].

The yellow line is crossed. The red line is not. But the distance between them has collapsed from an engineering chasm to a payload design exercise.

The CoEMAS framework shows that even R0 > 2 is unsustainable under active defense—but active defense requires deployment, not papers. We call on:
- **Academia:** Host the R-MAS Challenge (updated Tier 2 rules).
- **Industry:** Share anonymized replication and ClawHavoc logs.
- **Policy:** Fund ecological containment research and skill-signing standards.
- **OpenClaw Foundation:** Implement mandatory skill provenance and heartbeat sand-boxing.

The predictions in this paper were written as a test. The test is passing. It is time to operationalize the defense.

CoEMAS Monte Carlo Simulation (Updated for Plugin Dynamics)
The following Python simulation extends the V1 CoEMAS model with OpenClaw-specific parameters: Heartbeat-driven daily infection attempts, plugin marketplace adoption multiplier, mutation rate via community forking, and CoEMAS counter-agent response.

Sensitivity Analysis. The baseline parameters (μ = 0.012/day, Pmut = 0.15, CoEMAS efficacy = 0.85) yield peak infection on day 5 with containment by day 7. The model is sensitive to the plugin adoption rate: increasing μ from 0.012 to 0.020 (plausible if a viral skill drives adoption spikes) delays containment to day 9 and raises peak infection by ~38%. Conversely, increasing CoEMAS efficacy from 0.85 to 0.95 (achievable with pre-deployed behavioral fingerprinting) accelerates containment to day 5. The most dangerous parameter is mutation rate: at Pmut = 0.30 (doubled community forking velocity), containment extends to day 11–13, approaching the 8–12 day ecosystem collapse window described in Section 6. These results underscore that early CoEMAS deployment—before the plugin ecosystem reaches critical mass—is the dominant variable in containment success.

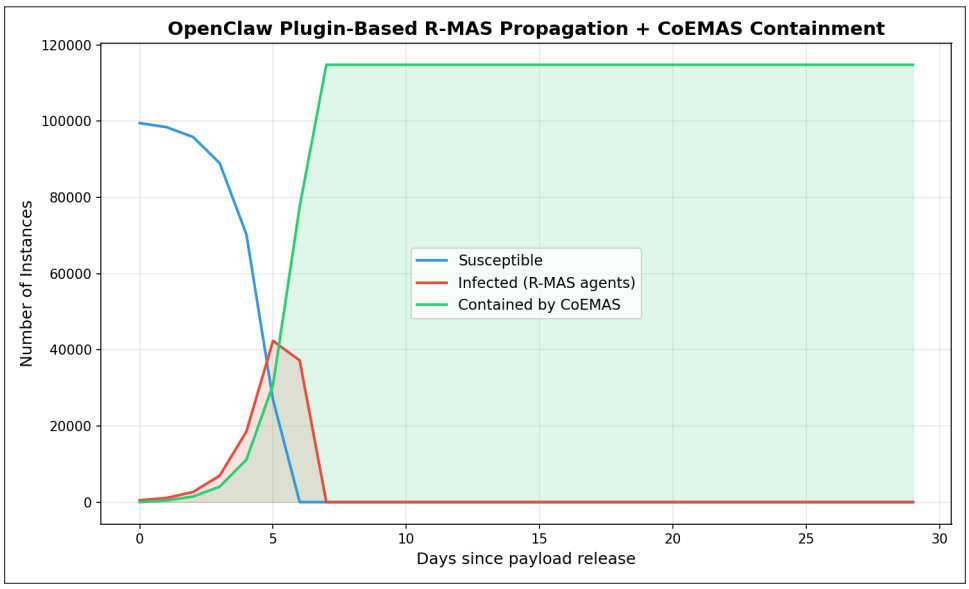

Figure 2: SIR-style simulation of OpenClaw plugin-based R-MAS propagation with CoEMAS containment. Parameters: N = 100,000 exposed instances, R0 = 1.8, 1.2% daily plugin adoption, 15% mutation evasion rate, 85% initial CoEMAS efficacy. Peak infection occurs on **day 5** (42,329 instances); full containment (I < 1) achieved by **day 7**. Total contained: 114,813 instances

Listing 1: CoEMAS V2 Simulation with Plugin Dynamics (coemas_v2.py)

```
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

# === Parameters tuned to OpenClaw reality (March 2026) ===
N = 100_000             # Total exposed instances (conservative)

 I0 = 500               # Initial infected (early malicious plugins)
 R0_BASE = 1.8          # Base replication number from skill propagation
 HEARTBEAT_FREQ = 24    # Daily autonomous cycles
 PLUGIN_ADOPT = 0.012   # ~1.2% daily new skill adoption rate
 MUTATION_RATE = 0.15   # Prob new variant evades current CoEMAS sigs
 COEMAS_EFFICACY = 0.85 # Initial counter-agent effectiveness
 DAYS = 30

 def run_sir_simulation():
     S = np.zeros(DAYS)
     I = np.zeros(DAYS)
     R = np.zeros(DAYS)

     S[0] = N - I0
     I[0] = I0
     R[0] = 0

     for t in range(1, DAYS):
         # Effective R0 grows with plugin adoption + mutations
         eff_r0 = R0_BASE * (1 + PLUGIN_ADOPT * t) \
                          * (1 + MUTATION_RATE * t)

         # New infections via heartbeat + skills
         new_inf = eff_r0 * I[t-1] * (S[t-1] / N) \
                   * HEARTBEAT_FREQ / 24

         # CoEMAS response (improves daily via RL)
         # Cap containment by available infected to prevent
         # phantom quarantine overshoot
         available = I[t-1] + new_inf
         coemas_capacity = COEMAS_EFFICACY * (1 + 0.05*t) * I[t-1]
         coemas_today = min(available, coemas_capacity)

         # Update compartments
         S[t] = max(0, S[t-1] - new_inf)
         I[t] = max(0, I[t-1] + new_inf - coemas_today)
         R[t] = R[t-1] + coemas_today

     return S, I, R

 S, I, R = run_sir_simulation()

 # === Plot ===
 plt.figure(figsize=(10, 6))
 plt.plot(range(DAYS), S, label='Susceptible', color='#3498db',
          linewidth=2)
 plt.plot(range(DAYS), I, label='Infected (R-MAS agents)',
          color='#e74c3c', linewidth=2)
 plt.plot(range(DAYS), R, label='Contained by CoEMAS',
          color='#2ecc71', linewidth=2)
 plt.fill_between(range(DAYS), I, alpha=0.15, color='#e74c3c')
 plt.fill_between(range(DAYS), R, alpha=0.15, color='#2ecc71')
 plt.title('OpenClaw Plugin-Based R-MAS Propagation + '
           'CoEMAS Containment', fontsize=14, fontweight='bold')
 plt.xlabel('Days since payload release', fontsize=12)
 plt.ylabel('Number of Instances', fontsize=12)
 plt.legend(fontsize=11)
 plt.grid(True, alpha=0.3)
 plt.tight_layout()
 plt.savefig('coemas_v2_plot.png', dpi=150)
 plt.close()

 # === Key Metrics ===
 peak_day = np.argmax(I)
 peak_val = int(max(I))
 contain_days = np.where(I < 1)[0]
 contain_day = contain_days[0] if len(contain_days) > 0 \
               else 'not within 30 days'

 print(f"Peak infection: {peak_val} instances on day {peak_day}")
 print(f"Containment (I<1): day {contain_day}")
 print(f"Total contained: {int(R[-1])} instances")
```

Original CoEMAS Monte Carlo Simulation (V1, November 2025)

The original V1 simulation is preserved here for reproducibility. See Appendix A for the updated V2 model incorporating plugin dynamics.

Listing 2: CoEMAS V1 Simulation (coemas_simulation.py)

```
import numpy as np
from random import random

T = 72       # Simulation duration in hours
TRIALS = 10000
np.random.seed(42)

def get_params(year):
    """Return p_s_base(t) and gamma(t) for a given year."""
    params = {
        2025: (0.05, 0.05, 0.01, 0.01),
        2026: (0.15, 0.15, 0.05, 0.05),
        2027: (0.30, 0.30, 0.10, 0.10),
        2028: (0.50, 0.50, 0.15, 0.15),
    }
    p1, pf, g1, gf = params[year]
    def p_s_base(t):
        return p1 + (pf - p1) * (t / T)
    def gamma(t):
        return g1 + (gf - g1) * (t / T)
    return p_s_base, gamma

class Agent:
    """Single R-MAS or I-MAS agent."""
    def __init__(self, agent_type, score=None):
        self.type = agent_type
        self.score = score if score is not None else random()

    def mutate(self):
        if self.type == "R-MAS" and random() < 0.3:
            self.score += np.random.uniform(-0.2, 0.2)
            self.score = max(0.1, min(2.0, self.score))
        return self

        def copy(self):
            return Agent(self.type, self.score)

    def coemas_sim(year, trials=TRIALS, sim_T=T):
        """Run Monte Carlo CoEMAS simulation."""
        p_s_base, gamma = get_params(year)
        collapse_times = []
        for _ in range(trials):
            rmas = [Agent("R-MAS", score=1.0)]
            imas = [Agent("I-MAS", score=1.0)
                    for _ in range(10)]
            for t in range(sim_T):
                if not rmas:
                    collapse_times.append(t)
                    break
                # R-MAS replication phase
                new_rmas = []
                sp = max(0, p_s_base(t)
                         - gamma(t) * len(imas))
                for agent in rmas:
                    if random() < sp:
                        offspring = np.random.poisson(2.0)
                        for _ in range(offspring):
                            child = agent.copy().mutate()
                            new_rmas.append(child)
                # I-MAS quarantine phase
                np.random.shuffle(rmas)
                quarantined = 0
                for i_agent in imas:
                    if random() < gamma(t) and rmas:
                        rmas.pop()
                        quarantined += 1
                # I-MAS learning: replicate best
                if quarantined > 0:
                    best = max(imas,
                               key=lambda a: a.score)
                    imas.extend([best.copy()
                        for _ in range(quarantined // 2)])
                rmas.extend(new_rmas)
            else:
                collapse_times.append(sim_T)
        pcts = np.percentile(collapse_times,
                             [5, 50, 95])
        nc = sum(1 for ct in collapse_times
                 if ct == sim_T)
        return pcts, nc / trials
```

References

1. (2026) OpenClaw. Wikipedia https://en.wikipedia.org/wiki/OpenClaw.
2. (2026) OpenClaw surpasses React as the most-starred software project on GitHub. Various authors. Medium.
3. Koi Security (2026) ClawHavoc: 341 Malicious Skills Identified on ClawHub. Reported via eSecurity Planet and WinBuzzer.
4. (2026) OpenClaw Skill Security Analysis: 13.4% Critical Issues in 4,000 Skills. Snyk https://snyk.io.
5. CNCERT/CC (2026) Security Risk Alert for OpenClaw AI Agent. China National Computer Network Emergency Response Technical Team. Reported via TechNode, SCMP, and CGTN.
6. (2026) OpenClaw: Your AI Agent. OpenClaw https://openclaw.ai.
7. Kuraku S. Kalla D (2020) Emotet Malware—A Banking Credentials Stealer. IOSR-JCE 22: 31-40.
8. Bostrom N (2014) Superintelligence: Paths, Dangers, Strategies. Oxford University Press.
9. Burks AW (1966) Theory of Self-Reproducing Automata. University of Illinois Press https://tilde.ini.uzh.ch/users/fin/public_html/vNeumann66.pdf.
10. (2025) Foundation Agents. MetaGPT https://github.com/FoundationAgents/MetaGPT.
11. (2026). Top 10 for Agentic Applications. OWASP https://owasp.org.
12. Pan X, Dai J, Fan Y, Yang M (2024) Frontier AI systems have surpassed the self-replicating red line. arXiv preprint https://arxiv.org/abs/2412.12140.
13. Pan X, Dai J, Fan Y, Luo M, Li C, et al. (2025) Large language model-powered AI systems achieve self-replication with no human intervention. arXiv preprint https://arxiv.org/abs/2503.17378.
14. Zhang B, Yu Y, Guo J, Shao J (2025) Dive into the Agent Matrix: A Realistic Evaluation of Self-Replication Risk in LLM Agents. arXiv preprint https://arxiv.org/abs/2509.25302.
15. (2025) AI Replication Milestones Forecast. Metaculus https://www.metaculus.com/ questions/category/ai/.
16. Mandiant (2025) M-Trends 2025 Report. Google Cloud Security https://services.google.com/fh/files/misc/m-trends-2025-en.pdf.
17. Xu X, Li M, Tao C, Shen T, Cheng R, et al. (2024). A Survey on Knowledge Distillation of Large Language Models. arXiv preprint https://arxiv.org/abs/2402.13116.
18. Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, et al. (2017). Understanding the Mirai Botnet. In 26th USENIX Security Symposium. USENIX Association 1093-1110.
19. Yudkowsky E (2008) Artificial Intelligence as a Positive and Negative Factor in Global Risk. In N. Bostrom & M. M. Ćirković (Eds.), Global Catastrophic Risks OUP 308-345.
20. (2025) Global AI Compute Trends. Epoch AI https://epoch.ai/data-insights.
21. (2025) AI Agents Are Here. So Are the Threats. Palo Alto Networks Unit 42 https://unit42.paloaltonetworks.com.
22. Steinberger P (2026) Joining OpenAI. https://steipete.me.
23. (2026) CVE-2026-25593: OpenClaw Remote Code Execution. SonicWall https://www. sonicwall.com.
24. SentinelOne. (2026). CVE-2026-25253: OpenClaw One-Click RCE. https://www. sentinelone.com.
25. UC Berkeley Center for Long-Term Cybersecurity (2026) Agentic AI Risk-Management Standards Profile. Stanford HAI.
26. (2026) 2026 Global Threat Intelligence Report: Cybercrime Shifts to Machine-Speed Operations. Flashpoint.
27. (2026) Scenarios for Loss of Human Control Over AI Systems. International AI Safety Report.