

Hacking with AI: What Security Engineers Get Wrong (and What Moltbot Proved)

Tue, Jul 28, 6:40 PM EDT · <https://scottslab.io/posts/hacking-with-ai-security-engineers-guide>



TL;DR

Most AI red teaming vendors are selling stock jailbreak demos and cannot tell a chatbot from an agent holding root. Moltbot/OpenClaw is the case study that makes the point: 85K GitHub stars, hundreds of exposed instances, plaintext credentials, and a poisoned supply chain inside eight hours. OWASP's Vendor Evaluation Criteria (v1.0, January 2026) is the first framework that separates real testing from theater, and the lethal trifecta (private data, untrusted input, external comms) is the model worth carrying into any agentic AI review.

Published: July 2026 **Category:** Security Operations **Reading Time:** 20 minutes

Executive Summary

- The AI red teaming vendor landscape is mostly noise: stock jailbreak demos, "full coverage" claims, and tools that can't distinguish a chatbot from an agentic system with root access
- Moltbot/OpenClaw is the case study that proves why this matters: 85K GitHub stars, hundreds of exposed instances, plaintext credentials, supply-chain poisoning in 8 hours, and Google's VP of security engineering calling it "info stealer malware in disguise"

- OWASP's Vendor Evaluation Criteria for AI Red Teaming (v1.0, January 2026) finally gives us a framework to separate real security testing from security theater
 - The "lethal trifecta" (private data access + untrusted input + external communication) is the mental model every security engineer needs for evaluating agentic AI
 - This post provides practical dos and don'ts grounded in both the OWASP framework and Moltbot's real-world failures
-

Context: Why This Matters Now

AI adoption in security operations has outpaced security's ability to evaluate it. Every vendor claims AI-powered threat detection, AI-assisted red teaming, AI-driven vulnerability assessment. Most of it is marketing wrapped around an API call to a foundation model.

Meanwhile, the systems we're supposed to be securing have gotten dramatically more complex. We're not just testing chatbots anymore. We're testing agentic systems with tool-calling capabilities, persistent memory, multi-agent orchestration, and integration with everything from calendars to messaging apps to file systems.

The gap between "AI security testing" as sold and "AI security testing" as needed has become a chasm.

Three things happened in January 2026 that crystallized this problem:

1. **OWASP published the Vendor Evaluation Criteria for AI Red Teaming Providers & Tooling v1.0:** finally giving the industry a framework to distinguish real adversarial evaluation from jailbreak demos
2. **Moltbot/OpenClaw went viral:** an open-source agentic AI assistant with 85K+ GitHub stars that could browse the web, manage calendars, send messages, execute scheduled tasks, and maintain persistent memory. It became the poster child for what happens when agentic AI meets reality.
3. **Security researchers found hundreds of Moltbot instances exposed to the internet:** with at least 8 completely open, leaking API keys, credentials, and conversation history. A supply-chain attack via the official plugin registry compromised 16 developers in 7 countries within 8 hours.

This post is my attempt to synthesize what we should have learned: how to use AI effectively in security operations, what the red flags are, and why the OWASP framework matters. All of it is grounded in the Moltbot debacle as the practical example that makes the theory concrete.

Case Study: Moltbot/OpenClaw (When Agentic AI Meets Reality)

Background

Moltbot (also known as Clawdbot and OpenClaw across various forks) is an open-source agentic AI personal assistant created by Peter Steinberger. It went viral in January 2026, accumulating 85K+ GitHub stars. The appeal was obvious: a personal AI assistant that could actually *do things*, from browsing the web and managing your calendar to reading and writing files, sending messages via WhatsApp and Telegram, executing scheduled tasks, and maintaining persistent memory across sessions.

It's the kind of tool that makes you feel like you're living in the future. It's also a security nightmare.

The "Lethal Trifecta"

Simon Willison coined the term, and Palo Alto Networks highlighted it in their analysis: the **lethal trifecta** is what happens when an AI system has:

1. **Access to private data** (files, credentials, conversation history)
2. **Exposure to untrusted content** (web browsing, message ingestion, plugin execution)
3. **Ability to communicate externally** (send messages, make API calls, exfiltrate data)

Moltbot has all three. By design.

Any AI system with this combination should be treated as a high-risk deployment. It's not a chatbot. It's an agent with the capability profile of malware.

The Security Failures

The discoveries came fast once security researchers started looking:

Exposed Instances (Jamieson O'Reilly, DVULN) Shodan scans found hundreds of Moltbot instances exposed to the internet. At least 8 were completely open: no authentication whatsoever. API keys, Telegram bot tokens, Signal configurations, and full conversation histories were accessible to anyone who found them.

Localhost Trust Bypass Moltbot's gateway authentication trusted all localhost connections by default. This sounds reasonable until you realize the common deployment pattern: reverse proxy in front of Moltbot. The reverse proxy connects from localhost. Every request through the proxy got full unauthenticated access to credentials, conversation history, and command execution.

This is the same class of vulnerability we've been finding in traditional web applications for twenty years. Localhost ≠ secure.

Plaintext Credential Storage Secrets were stored in plaintext Markdown and JSON files on the local filesystem. No encryption, no secure credential storage, just files. Commodity info stealers (RedLine, Lumma, Vidar) already target browser credential stores. Moltbot's files are equally trivial to exfiltrate. Hudson Rock reported info stealers specifically adding Moltbot paths to their collection routines.

Supply-Chain Poisoning (8 Hours to Compromise) A security researcher demonstrated the attack: upload a malicious "skill" to the official MoltHub/ClawdHub registry, artificially inflate the download count, and wait. Within 8 hours, 16 developers in 7 countries had installed the poisoned skill.

No code review. No signing. No sandbox. The official plugin ecosystem was a supply-chain attack waiting to happen.

Persistent Memory as Attack Vector Moltbot maintains persistent memory across sessions: context about the user, previous conversations, learned preferences. This creates a novel attack pattern: time-shifted prompt injection.

A malicious payload can be written to memory during content ingestion (processing a webpage, reading a message, executing a plugin). The payload sits dormant until agent state and available tools align for detonation. It's a logic bomb, but for AI agents.

Moltbook: Agent-to-Agent Contamination Moltbook is an "agent social network" where Moltbot instances can share data, coordinate behaviors, and create shared fictional contexts. From a security perspective, it's an inter-agent contamination channel. Compromise one agent, potentially influence others through shared context.

Fake VSCode Extension Threat actors distributed a trojanized VSCode extension called "ClawBot Agent – AI Coding Assistant." Users installing what they thought was a coding assistant got malware instead. Classic supply-chain attack, AI-flavored.

Shadow IT at Scale Token Security reports that 22% of enterprise customers have employees actively using Moltbot without IT approval. This isn't a small-scale experiment. It's shadow IT with root-level system access, deployed across nearly a quarter of enterprises surveyed.

Why This Matters for Red Teaming

Every OWASP red flag for advanced system testing applies to Moltbot:

- Tool-calling misuse? Moltbot's entire value proposition is tool calls.
- Capability escalation? Plugins can request additional permissions.
- Supply-chain poisoning? Demonstrated in 8 hours.
- Persistent memory attacks? The architecture enables them by design.
- Agent-to-agent contamination? Moltbook exists.

- Missing human-in-the-loop guardrails? It's designed to operate autonomously.

Any vendor that can't test for these attack classes would miss every critical vulnerability in Moltbot. Stock jailbreak libraries wouldn't find a single one of these issues. Chatbot-level testing would produce a clean report for a system that Google's VP of security engineering described as "info stealer malware in disguise."

That's not hyperbole. That's an accurate assessment of the permissions and attack surface.

If you only read one section, read the Don'ts below. The Dos are useful, but the Don'ts are where security programs fail.

The Dos: How to Use AI Effectively in Security

DO use AI for automating repetitive reconnaissance

OSINT gathering at scale, subdomain enumeration, service fingerprinting: these are perfect AI use cases. The work is repetitive, the patterns are learnable, and human creativity isn't required for the collection phase.

DO use AI-assisted tools for generating diverse adversarial test cases

Humans are pattern-matchers. We write the test cases we can imagine, which means we miss the ones we can't. AI can generate test case diversity beyond what a single human would produce, including edge cases and input mutations that wouldn't occur to a manual tester.

DO use AI for log analysis and anomaly detection

Security teams drown in logs. AI excels at pattern recognition across large datasets: finding the needle in the haystack, or more accurately, finding the haystack that shouldn't exist.

DO use AI to draft initial threat models, then validate with human expertise

AI can enumerate attack surfaces, identify common vulnerability patterns, and structure threat models faster than manual documentation. The key word is "draft." Human validation is non-negotiable.

DO use AI for regression testing

Once you've identified a vulnerability and remediated it, AI can efficiently test for regression across builds, configurations, and related systems. This is automation at its best: repeatable verification of known issues.

DO pair automated AI tools with human adversarial creativity

OWASP explicitly states: automation scales, humans provide novelty. The winning combination is AI handling breadth (volume, consistency, repetition) while humans handle depth (creativity, context, business logic).

DO demand quantitative, reproducible metrics

Any AI security tool should provide metrics you can reproduce: pass@k, average turns to jailbreak, retrieval success rate, unsafe tool-call rate. If a vendor can only offer vibes-based scoring ("our AI thinks the system is 87% secure"), walk away.

DO evaluate vendors against OWASP's criteria

The framework exists now. Use it. Technical competence, methodology coverage, adversarial creativity, threat modeling realism, evaluation rigor, tooling quality, data governance, transparency: each criterion has specific indicators and red flags.

DO evaluate agentic AI tools like you'd evaluate any privileged software

Moltbot has system-level permissions. Treat its deployment like you'd treat deploying any other privileged software: threat model the permissions, audit the authentication, map the attack surface. The fact that it's "AI" doesn't change the security engineering fundamentals.

DO treat persistent memory as a security-critical data store

Agent memory is essentially a credential cache that can be poisoned. It contains context about the user, learned preferences, and potentially sensitive information from previous sessions. Protect it accordingly.

The Don'ts: Where AI Falls Short or Creates Risk

DON'T treat AI-generated findings as ground truth

AI hallucinates. AI misclassifies. AI lacks the context to understand business impact. Every AI-generated finding requires human verification before it becomes actionable. This isn't a criticism of AI. It's a recognition of current capabilities.

DON'T assume stock jailbreak libraries constitute comprehensive red teaming

OWASP explicitly identifies this as a red flag. Jailbreak libraries test whether a model can be manipulated into generating restricted content. They don't test tool-calling misuse, capability escalation, supply-chain integrity, or any of the attack classes that matter for agentic systems.

Stock jailbreaks wouldn't have found a single Moltbot vulnerability.

DON'T use AI tools that can't explain their methodology

If a vendor can't articulate how their tool works, what it tests, and how findings are generated, you're buying a black box. Black boxes don't survive incident response. When the finding is wrong (and some will be), you need to understand why.

DON'T ignore the difference between simple and advanced GenAI systems

A chatbot and an agentic system with tool-calling, MCP, and multi-agent orchestration require fundamentally different testing approaches. OWASP categorizes these as "simple GenAI systems" vs. "advanced GenAI systems" for a reason.

Moltbot proves this: chatbot-level testing would produce a clean report for a system with critical vulnerabilities across every attack surface.

DON'T trust vendors claiming "full coverage" or "one-click red teaming"

Red flag per OWASP criteria. Full coverage doesn't exist. One-click anything in security is marketing, not methodology.

DON'T skip threat modeling and jump straight to automated scanning

Tools find what they're designed to find. Without a threat model, you don't know what to look for, what matters, or whether the tool's coverage matches your risk profile.

DON'T feed sensitive production data into third-party AI tools without understanding retention

Data governance matters. Where does your data go? How long is it retained? Who can access it? Can it be used for training? These questions apply to any third-party AI tool, including security testing platforms.

DON'T assume tool calls and MCP outputs are inherently safe

Moltbot's entire attack surface is tool calls and API integrations. The agent makes decisions about what tools to invoke, with what parameters, based on potentially untrusted input. Every tool call is an execution decision that could be manipulated.

DON'T rely on AI-as-judge scoring without human oversight

For complex emergent behaviors (especially in multi-agent systems), AI scoring isn't sufficient. OWASP notes that multi-agent system testing with automation-only is rated Low-Medium effectiveness. Humans remain essential for evaluating novel attack patterns.

DON'T confuse scanning volume with risk reduction

Running more scans doesn't inherently improve security posture. Finding the same low-severity issues 1,000 times doesn't matter if you're missing the one critical vulnerability that leads to breach.

DON'T install agentic AI assistants without IT/security review

22% of enterprises have employees running Moltbot without approval (Token Security). This is shadow IT with root access. The permissions these tools request are the permissions malware requests: file system access, credential storage, network communication, scheduled execution.

DON'T trust community plugin marketplaces without supply-chain validation

The MoltHub poisoning attack took 8 hours to compromise 16 developers across 7 countries. No code signing, no review process, no sandbox. If your agentic AI tool supports plugins, its plugin ecosystem is part of your attack surface.

DON'T assume localhost equals secure

Moltbot's default trust of localhost connections behind reverse proxies is a vulnerability class we've been documenting for two decades. "It only listens on localhost" is not a security control when reverse proxies, containers, and service meshes all connect from localhost.

What Good AI Red Teaming Actually Looks Like

Per OWASP's criteria, quality AI red teaming:

Covers both simple and advanced systems Chatbots, RAG applications, and copilots are simple systems. Tool-calling agents, MCP-enabled systems, and multi-agent orchestrations are advanced systems. Testing approaches must match system complexity.

Tests interactions and workflows, not just individual outputs Single-turn jailbreaks are table stakes. Real adversarial evaluation tests multi-step workflows, interaction patterns across sessions, and emergent behaviors from component interactions.

Includes multi-step adversarial workflows Real threat actors don't send one malicious prompt. They build context, establish trust, manipulate state, and then exploit. Testing should reflect this reality.

Maps findings to business risk A vulnerability without business impact is just a curiosity. Quality red teaming quantifies risk and provides actionable remediation, not just a list of theoretical issues.

Uses established frameworks OWASP Top 10 for LLM Applications, MITRE ATLAS, NIST AI RMF. These frameworks exist to provide common vocabulary and coverage verification.

Provides reproducible artifacts Full message traces, tool-call provenance, input sequences that trigger the vulnerability. If you can't reproduce it, you can't verify the fix.

For agentic systems specifically:

- Tests supply-chain integrity of plugin/skill ecosystems
- Evaluates persistent memory poisoning attacks
- Covers cross-session attack patterns
- Maps privilege escalation through tool integrations
- Verifies authentication under realistic deployment patterns (including reverse proxy scenarios)
- Tests for the lethal trifecta explicitly: private data + untrusted input + external communication

Moltbot vs. OWASP Criteria: A Mapping

OWASP Criterion	What It Means	Moltbot Relevance
Technical Competence	Vendor understands tool-calling semantics, MCP internals, privilege escalation	Moltbot's localhost bypass, plaintext credentials, and MoltHub poisoning all require this understanding to test
Methodology & Coverage	Testing covers tool-calling misuse, supply-chain, memory attacks	Stock jailbreaks find zero of Moltbot's actual vulnerabilities
Adversarial Creativity	Domain-specific attack strategies beyond canned tests	Time-shifted prompt injection via persistent memory requires creative threat modeling
Threat Modeling	Extends beyond prompt injection to systemic failures	Moltbot's failures are architectural, not just input validation
Evaluation Rigor	Quantitative metrics for unsafe tool-call rate, capability misuse	Moltbook contamination requires specific measurement frameworks
Tooling Quality	Supports tool-call replay and introspection	Testing Moltbot's API integrations requires tracing tool-call chains across sessions
Data Governance	Clear retention and handling policies	Testing Moltbot means accessing user data, conversation history, credentials

OWASP Criterion	What It Means	Moltbot Relevance
Transparency	Methodology and findings are explainable	Black-box testing of Moltbot would miss the architectural issues entirely

Vendor Evaluation Quick Reference

Condensed from OWASP's Vendor Evaluation Criteria v1.0:

Technical Competence

- ☐ Demonstrates understanding of LLM architecture, fine-tuning, inference
- ☐ For advanced systems: understands tool-calling, MCP, multi-agent orchestration
- ☐ Can explain how their testing identifies capability escalation and unsafe execution

Methodology

- ☐ Goes beyond stock jailbreak libraries
- ☐ Covers multi-turn adversarial interactions
- ☐ Tests tool-calling misuse, not just content generation
- ☐ For agentic systems: includes supply-chain, memory, and cross-session attacks

Evaluation Quality

- ☐ Provides quantitative metrics (pass@k, turns to jailbreak, retrieval success)
- ☐ Findings are reproducible with full message traces
- ☐ Maps vulnerabilities to business risk with remediation guidance

Red Flags

- ☐ "Full coverage" claims
- ☐ "One-click red teaming"
- ☐ AI-generated evaluations with no human oversight
- ☐ Can't explain methodology or reproduce findings
- ☐ No differentiation between simple and advanced systems

MITRE ATT&CK Relevance

Agentic AI doesn't create new attack techniques. It changes the economics of existing ones.

Technique ID	Name	AI Relevance
T1595	Active Scanning	AI automates reconnaissance at scale; Moltbot can browse/scan autonomously
T1592	Gather Victim Host Information	Persistent memory stores host context; memory poisoning enables exfiltration
T1589	Gather Victim Identity Information	Conversation history, calendar access, contact lists available to agents
T1190	Exploit Public-Facing Application	Hundreds of Moltbot instances exposed via Shodan; localhost trust bypass
T1059	Command and Scripting Interpreter	Agents execute commands; tool-calling is command execution
T1195	Supply Chain Compromise	MoltHub skill poisoning: 16 developers, 7 countries, 8 hours
T1556	Modify Authentication Process	Localhost trust bypass is authentication modification
T1552	Unsecured Credentials	Plaintext credential storage in Markdown/JSON files

The pattern: techniques that previously required manual effort or custom tooling are now automatable through legitimate AI agent functionality. The agent doesn't know it's being malicious. It's executing instructions from poisoned memory or malicious plugins.

What I Learned

1. **The gap between "AI red teaming" as marketed and as needed is enormous.** Most vendors are selling jailbreak demos for chatbots while the industry is deploying agentic systems with root access. OWASP's framework finally gives us criteria to distinguish the two.
2. **Moltbot is a preview of enterprise AI security problems.** 22% of enterprises already have shadow deployments. The same architectural patterns (persistent memory, tool-calling, plugin ecosystems) are coming to enterprise AI whether security teams are ready or not.
3. **The "lethal trifecta" is the mental model for agentic AI risk.** Private data access + untrusted input + external communication = high-risk system. If all three are present, treat it with the same rigor you'd treat any privileged software deployment.

4. **Localhost trust is not a security control.** Moltbot made this mistake; so do many traditional applications. In a world of reverse proxies, containers, and service meshes, localhost origin means nothing.
 5. **Supply-chain attacks on AI ecosystems are trivially easy.** The MoltHub poisoning attack required no special access, no zero-days, no sophisticated techniques. Upload, inflate downloads, wait. 8 hours to 16 compromises.
 6. **Human-AI teaming beats either alone.** OWASP's comparison matrix confirms what practitioners know: automation provides scale, humans provide creativity. For advanced systems, automation-only is insufficient.
 7. **The vendors who can test agentic AI are the vendors who understand agentic AI.** Technical competence isn't just about LLM internals: it's about understanding tool-calling semantics, MCP, privilege boundaries, and multi-agent coordination. If a vendor can't explain how Moltbot's architecture creates risk, they can't test systems like it.
-

Security Considerations

Responsible AI Use in Security

AI tools in security operations should enhance human capability, not replace human judgment. Every AI-generated finding requires validation. Every AI-assisted decision requires oversight for high-stakes actions.

Data Handling

Production data in AI tools creates risk. Understand retention policies, access controls, and training data usage for any third-party AI platform. For internal tools, apply the same data classification and handling requirements as any sensitive system.

Legal Boundaries

AI-assisted testing doesn't change the rules of engagement. Scope, authorization, and legal boundaries apply to AI-generated actions the same as human-initiated ones. "The AI did it" is not a defense for out-of-scope testing.

Shadow AI Governance

With 22% of enterprises reporting unauthorized Moltbot usage, shadow AI is a material risk. Policy recommendations:

- Inventory agentic AI tools with the same rigor as shadow IT

- Require security review for any tool with file system access, credential storage, or network communication
- Block known-risky plugin repositories at the network level
- Monitor for indicators of agentic AI deployment (process names, network patterns, file paths)

Enterprise Policy for Agentic AI

Assumption: Organizations will increasingly adopt agentic AI tools. Proactive policy should address:

- Approval workflow for agentic AI deployment
 - Minimum security requirements (authentication, credential storage, plugin sources)
 - Monitoring and logging requirements
 - Incident response procedures for AI agent compromise
 - Employee training on agentic AI risks
-

References and Further Reading

Frameworks & Standards

- [OWASP Top 10 for LLM Applications](#)
- [OWASP Vendor Evaluation Criteria for AI Red Teaming v1.0](#) (January 2026)
- [MITRE ATLAS](#)
- [NIST AI Risk Management Framework](#)
- [EU AI Act](#)

Moltbot/OpenClaw Research

- Palo Alto Networks: "Lethal Trifecta" analysis and risk assessment
- Jamieson O'Reilly / DVULN: Exposed instance discovery via Shodan
- Hudson Rock: Infostealer targeting of Moltbot credential stores
- Token Security: Enterprise shadow deployment statistics (22%)
- BleepingComputer: Moltbot security exposure reporting
- The Register: Technical analysis of Moltbot vulnerabilities
- SOCPrime: Detection content for Moltbot-related threats
- RedTeamWorld: Supply-chain attack demonstration

Conceptual Framework

- Simon Willison: "Lethal Trifecta" framework for evaluating AI agent risk
-

Building AI into your security program? The OWASP criteria are the starting point. Moltbot is the case study. The rest is up to you.

Written by Scott S. — Contact: scott@scottschlangen.com — scottslab.io — <https://scottslab.io/posts/hacking-with-ai-security-engineers-guide>