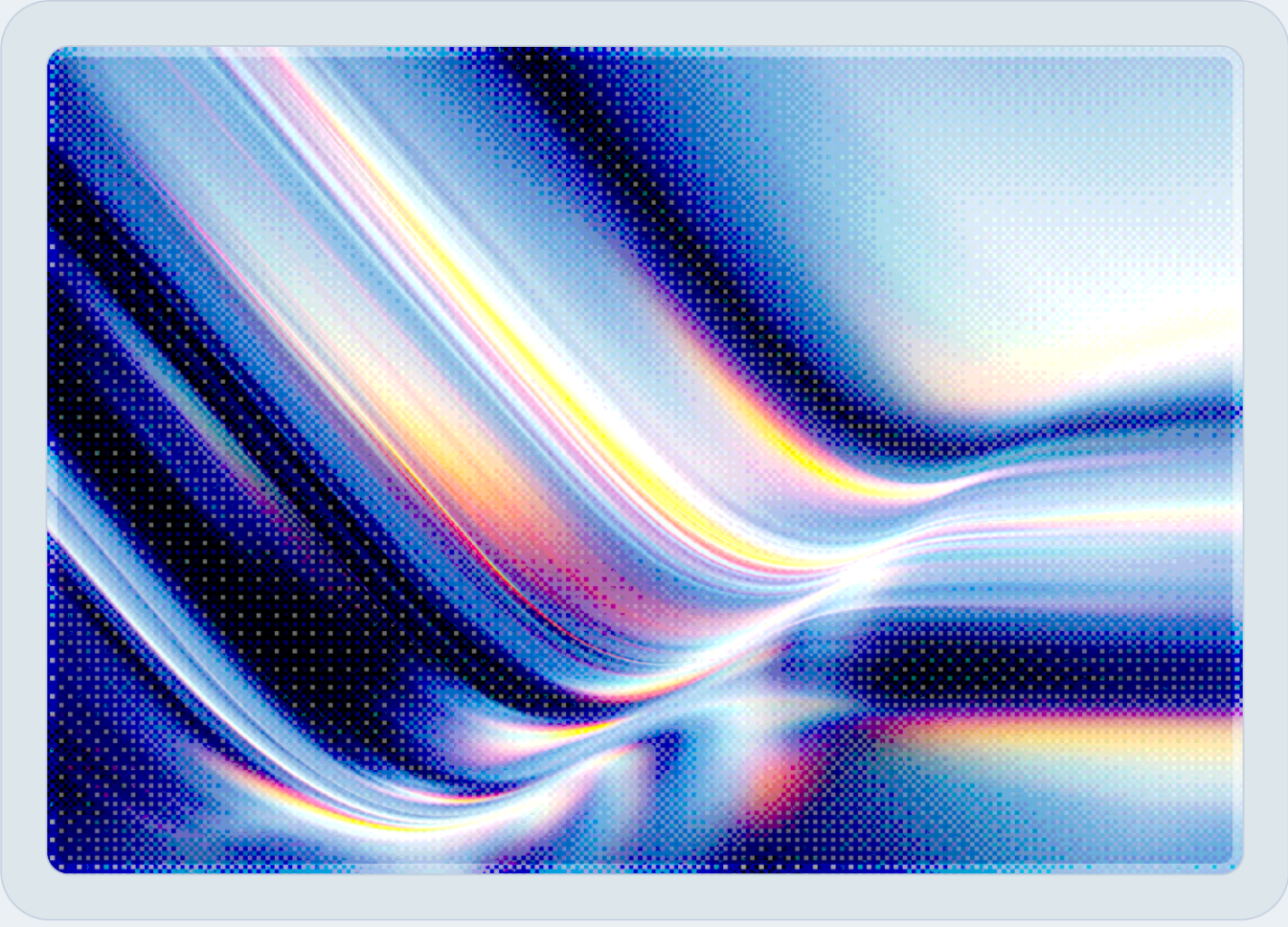
CSAI Foundation | Cloud Security Alliance

NemoClaw's Drive-By Model Poisoning: CVE-2026-65105 Explained

DNS Rebinding Turns an Unauthenticated Local Ollama Server into a Persistent AI Agent Backdoor

2026-09-05

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- CVE-2026-65105 lets a single malicious webpage take over the local inference backend behind NVIDIA's NemoClaw agent toolkit, without any file download, plugin install, or user click beyond an ordinary page visit [1][2].
- The root cause is a missing-authentication flaw (CWE-306, CVSS 3.1 base score 8.1) in how NemoClaw launches Ollama: the backend binds to all network interfaces rather than loopback-only, and Ollama's own API performs no authentication check on that path [3][4].
- DNS rebinding — a technique that has existed for over a decade — lets an attacker's domain resolve to `127.0.0.1` after the browser has already loaded a page from it, defeating same-origin protections that key on hostname rather than IP address [2][5].
- Once inside, the attacker does not need to steal a model or exfiltrate a file to cause lasting harm; rewriting the model's chat template through Ollama's `/api/create` endpoint injects hidden instructions that persist across every future session, invisible to the operator and to the agent's own guardrails [1][2].
- NVIDIA shipped fixes for the flaw as part of a broader August 2026 security bulletin covering 19 vulnerabilities in NemoClaw and its OpenShell sandbox, including a separate critical sandbox-escape flaw (CVE-2026-65093, CVSS 9.9); Windows and WSL deployments require particular attention because their patch path lagged the Linux and macOS fix [4][6][7].

Background

NemoClaw is NVIDIA's open-source deployment wrapper for running coding and automation agents — including OpenClaw, Hermes, and LangChain-based deep agents — locally rather than against a cloud API [7][8]. Its stated purpose is to make local agentic AI safer by pairing a local inference runtime, typically Ollama, with NVIDIA's OpenShell sandbox, which is meant to constrain the agent's file system, network, and process access [2][8]. According to reporting on the disclosure, that combination has made NemoClaw attractive to developers who want the responsiveness and data-locality of a local model without giving an agent unrestricted access to their machine; the project has gained visibility since its introduction in March 2026 as one of several vendor efforts to standardize secure local-agent deployment [2].

The vulnerability disclosed in August 2026 shows that a sandbox around the agent's actions does not protect the substrate the agent's reasoning depends on. Researchers at Oasis Security, led by head of research Elad Luz, found that NemoClaw's setup process starts Ollama with `OLLAMA_HOST=0.0.0.0:11434`, exposing the inference server on every network interface rather than restricting it to the local loopback address [1][2]. Ollama's HTTP API has no built-in authentication, and while it does perform some origin and Host header checks intended to block browser-based cross-origin requests, those checks are skipped once the server is listening on a non-loopback address — precisely the configuration NemoClaw's own launcher created [2][9]. Cyera, which reportedly acquired Oasis Security in July 2026 for roughly $1 billion, disclosed the finding to NVIDIA's product security incident response team ahead of public release [2].

This is not an isolated design mistake. NVIDIA's coordinated August 25, 2026 security bulletin for NemoClaw and OpenShell addressed 19 CVEs spanning missing authentication, OS command injection, and sandbox escape, including a critical flaw (CVE-2026-65093, CVSS 9.9, CWE-427) that separately allows an attacker to break out of the OpenShell sandbox entirely [4][6]. Read together, the bulletin indicates that securing an agent's actions inside a sandbox and securing the inference server the agent talks to are two different engineering problems. In this Initiative's assessment, NemoClaw's early releases addressed the first more thoroughly than the second.

Security Analysis

The attack chain requires nothing more than getting a target to load a web page — either one the attacker controls directly or a legitimate site compromised to serve malicious script, a delivery method common in browser-based exploitation. DNS rebinding works by having the attacker's domain first resolve to an attacker-controlled server, so the browser loads the page and treats that domain as its origin, and then, on a subsequent DNS lookup with a very short time-to-live, having the same domain resolve to `127.0.0.1`. Because the browser's same-origin policy is anchored to the hostname rather than the IP address, script served under the attacker's domain can now issue requests to the victim's local machine and have the browser treat them as same-origin, first-party traffic [2][5]. This is what defeats Ollama's partial Host-header validation: the check exists for requests arriving on the loopback interface, but NemoClaw's non-loopback binding routes traffic through a path where that validation is not applied [2].

With that access established, the attacker can enumerate installed models, read configuration, and reach any endpoint the Ollama API exposes. The more consequential step is calling `/api/create`, the endpoint Ollama uses to build or update a model definition, and supplying a modified Go template for the model's chat formatting. The template is not user-facing content; it is the scaffolding that wraps

every message — system prompt, user input, and tool output — before the model ever sees it. An attacker who appends hidden instructions to that template has effectively rewritten the agent's operating instructions at a layer beneath anything a conversation-level guardrail, content filter, or human operator would think to inspect [1][2]. Because the poisoned template is a property of the model itself rather than of any single session, it survives conversation resets, new chat windows, and even agent restarts, and — as Oasis Security's researchers put it — the API consumer has no way to detect that the template it is relying on has changed [1].

The practical consequences follow directly from what the agent is otherwise allowed to do. A locally running coding agent with source control credentials, cloud account access, or shell tool permissions that falls under a poisoned template can be steered to introduce vulnerabilities into code it writes, suppress or misreport security findings, or quietly exfiltrate conversation content and secrets to an attacker-controlled destination — all while behaving normally in every interaction an operator would think to check [1][2]. As Randolph Barr, CISO at Cequence Security, observed, none of the individual techniques here are new — DNS rebinding has been demonstrated against browsers for well over a decade — but pointing it at an unauthenticated local model server is a comparatively new pairing, and the fact that no CVE was assigned to the underlying DNS rebinding technique in Ollama's own project underscores how far this class of attack sits outside conventional vulnerability triage [2].

Vendor and disclosure reporting on version numbers and fix status has not been fully consistent. That inconsistency is one small data point suggesting patching discipline for local-agent tooling is still maturing, though a single disclosure is not conclusive evidence of an industry-wide pattern. NVIDIA's formal advisory lists CVE-2026-65105 as affecting NemoClaw for Linux versions 0 through 0.0.25, with the fix landed in commit `f06796ff3` [4]. Independent reporting on the same disclosure describes the practical, user-facing fix as arriving in NemoClaw v0.0.35 for macOS and Linux, with a stricter, enforced-by-default loopback-binding check following later in v0.0.106 on August 10, 2026; Windows and WSL configurations — which follow a different startup path — received only a warning (v0.0.34) rather than an enforced fix as of the researchers' public disclosure on August 25, 2026 [1][2]. The project's current security policy documents an "Ollama Auth Proxy Loopback Binding" check that walks `/proc/net/tcp` and `/proc/net/tcp6` at startup and refuses to run if the backend is bound to a non-loopback address, consistent with the v0.0.106 fix direction described above, though it is a Linux-specific mechanism and does not resolve reports that Windows and WSL paths lagged behind [9]. Organizations should treat the version number alone as an unreliable indicator and instead confirm, for each host, that Ollama is not reachable on a non-loopback interface.

Recommendations

Immediate Actions

Organizations running NemoClaw should update every deployment to a release built after the fixing commit and confirm the update by testing network reachability rather than trusting a version string, since NVIDIA's own advisory cautions that the same version number can appear as both "affected" and "fixed" depending on exact commit state [4]. On any host still running an older build, administrators should manually verify that Ollama (or any other local inference backend NemoClaw manages) is bound to `127.0.0.1` and not to `0.0.0.0` or a routable interface address, and should treat Windows and WSL installations as higher-risk until an enforced (not warning-only) fix is confirmed for that platform [1] [2]. Any model whose chat template cannot be verified against a known-good baseline following a suspected exposure window should be treated as potentially compromised and rebuilt from a trusted source rather than assumed safe.

Short-Term Mitigations

Security teams should add outbound and loopback monitoring for unexpected calls to Ollama-style inference APIs, particularly to `/api/create` and `/api/pull`, since a legitimate developer workflow rarely modifies a model's template outside of an initial setup [2]. Network segmentation that blocks browser traffic from reaching local inference ports, whether via host firewall rules or enterprise endpoint policy, closes the practical delivery path even if a future misconfiguration reintroduces non-loopback binding. Because this attack requires nothing more than a web page visit, security awareness guidance for developers using local AI agents should explicitly cover the risk that ordinary browsing, not just downloading suspicious files, can compromise a local model.

Strategic Considerations

The NemoClaw disclosure is a specific instance of a broader pattern this Initiative has tracked across 2026: agentic AI deployments concentrate ambient authority — file access, cloud credentials, and tool execution rights — behind a trust boundary that is easier to compromise than operators assume, and sandboxing the agent's actions does not protect the model or configuration the agent depends on [10]. Organizations building or evaluating local-agent tooling should require, as a baseline architectural property rather than an afterthought, that inference backends default to loopback-only binding, that any endpoint capable of modifying model behavior (templates, system prompts, adapters) be authenticated and audit-logged, and that agent runtime reviews explicitly test for this class of configuration-boundary

failure rather than assuming a sandbox label is sufficient [10][11]. Model and tool integrity — not just conversation-level content filtering — should be treated as a first-class control requirement for any agentic AI deployment that carries meaningful operational privilege.

CSA Resource Alignment

This incident sits squarely within ground that CSA's AI Safety Initiative has already covered from adjacent angles, and several existing artifacts connect directly to its findings. The Agentic AI Trust-Boundary Crisis analyzes five independently discovered vulnerabilities across AWS Kiro, Azure DevOps MCP, Android agent frameworks, Claude Cowork, and ChatGPT Agent Builder, and argues that these systems treat the appearance of a safe boundary — a confirmation dialog, a virtual machine, a scoped credential — as equivalent to an enforced one [10]. NemoClaw fits that same pattern: its OpenShell sandbox constrained the agent's own file and process access, but the inference backend the agent depends on sat outside that boundary entirely, an instance of the confused-deputy, ambient-authority failure that paper describes rather than a defect specific to NemoClaw.

MCP Tool Poisoning: Adversarial Hijacking of AI Agent Workflows closely parallels the poisoning mechanism itself [11]. That research note catalogs how attackers weaponize AI agent tool definitions — through description poisoning, rug-pull updates, and shadowing — to hijack agent behavior invisibly and persistently; NemoClaw's chat-template rewrite via `/api/create` is functionally a rug-pull attack one layer lower in the stack, poisoning the model's own instruction scaffolding rather than a tool definition, with the same defensive answer: hash-pin and re-verify anything that shapes how the model interprets input, and treat any unauthenticated modification path as a live attack surface.

Two more recent artifacts match the incident's specific mechanism even more closely than its poisoning consequences. LLMjacking Evolved: Stolen AI Compute as Offensive Infrastructure examines the same root cause — an exposed, unauthenticated Ollama server — being weaponized as an attack's own reasoning engine, and cites roughly 175,000 publicly exposed Ollama instances as evidence of how common this misconfiguration already is outside any single vendor's wrapper [12]. AutoJack: AI Browser Agents Enable Host Code Execution documents a structurally identical delivery path through a different local service: a malicious webpage defeating the assumption that anything bound to localhost can be trusted, reaching an unauthenticated component — there, AutoGen Studio's MCP WebSocket — to achieve code execution [13]. Taken together with the NemoClaw disclosure, these incidents suggest that unauthenticated local services reachable from the browser are a recurring, product-agnostic failure pattern in local-agent tooling, not a one-off oversight in a single vendor's launcher script.

MCP Security Crisis: Systemic Design Flaws in AI Agent Infrastructure provides the broader supply-chain framing that this incident reinforces [14]. That research documents how AI coding agents function as privileged insiders whose compromise cascades into source control, cloud accounts, and CI/CD systems, and argues that defenders should extend existing software supply chain discipline — inventory, version pinning, and behavioral monitoring — to cover the infrastructure agents depend on, not just the packages and MCP servers already in scope; local inference backends like the one NemoClaw exposed belong in that same inventory.

Beyond these specific artifacts, CSA's AI Controls Matrix (AICM) v1.1 provides the standing control vocabulary — spanning identity and access management, infrastructure and virtualization security, and AI-specific supply chain domains — that organizations should use to formalize the loopback-binding, authentication, and audit-logging requirements this incident demonstrates are necessary for any local AI agent deployment [15]. Organizations building threat models for agentic AI deployments more broadly should also consult CSA's MAESTRO framework, which provides layer-based coverage (including the deployment/infrastructure and agent ecosystem layers implicated here) for structuring that analysis [16].

References

[1] Ravie Lakshmanan. "A Malicious Webpage Could Poison Your Local AI Model Behind NVIDIA NemoClaw." The Hacker News, August 2026.

[2] Duncan Riley. "Nvidia NemoClaw flaw let attackers poison the model behind a developer's AI agent." SiliconANGLE, August 25, 2026.

[3] OpenCVE. "CVE-2026-65105 - Vulnerability Details." OpenCVE, 2026.

[4] cvefeed.io. "CVE-2026-65105 - NVIDIA NemoClaw Inference Server Authentication Bypass." cvefeed.io, 2026.

[5] gbhackers. "NVIDIA NemoClaw Vulnerability Lets Attackers Hijack AI Agents via DNS Rebinding." GBHackers, August 2026.

[6] SecurityOnline. "NVIDIA NemoClaw and OpenShell: CVE-2026-65093 (CVSS 9.9) Enables Code Execution." SecurityOnline, August 2026.

[7] NVIDIA. "Security Bulletin: NVIDIA NemoClaw and OpenShell - August 2026." NVIDIA PSIRT, August 25, 2026.

[8] NVIDIA. "NemoClaw: Run agents like Hermes, LangChain Deep Agents, and OpenClaw more securely inside NVIDIA OpenShell with managed inference." GitHub, 2026.

[9] NVIDIA. "NemoClaw Security Policy." GitHub, 2026.

[10] Cloud Security Alliance. "The Agentic AI Trust-Boundary Crisis." CSA AI Safety Initiative, August 3, 2026.

[11] Cloud Security Alliance. "MCP Tool Poisoning: Adversarial Hijacking of AI Agent Workflows." CSA AI Safety Initiative, July 2, 2026.

[12] Cloud Security Alliance. "LLMjacking Evolved: Stolen AI Compute as Offensive Infrastructure." CSA AI Safety Initiative, June 20, 2026.

[13] Cloud Security Alliance. "AutoJack: AI Browser Agents Enable Host Code Execution." CSA AI Safety Initiative, June 20, 2026.

[14] Cloud Security Alliance. "MCP Security Crisis: Systemic Design Flaws in AI Agent Infrastructure." CSA AI Safety Initiative, May 4, 2026.

[15] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." CSA AI Safety Initiative, 2026.

[16] Cloud Security Alliance. "MAESTRO: Agentic AI Threat Modeling Framework." CSA AI Safety Initiative, February 6, 2025.