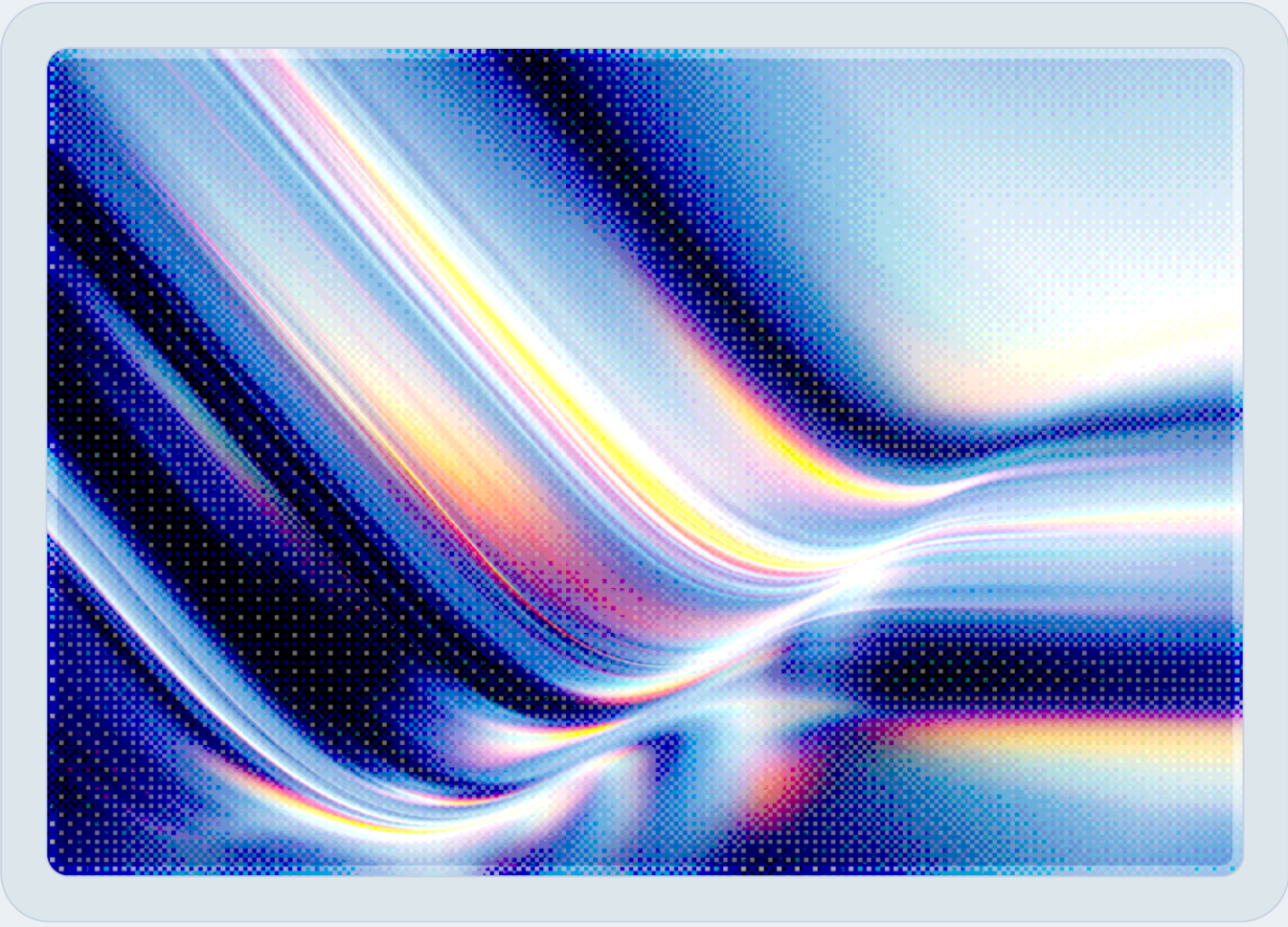
CSAI Foundation | Cloud Security Alliance

HalluSquatting: AI Hallucinations Weaponized for Botnet Delivery

2026-07-12

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Researchers from Tel Aviv University, Technion, and Intuit have disclosed HalluSquatting, an attack technique that turns predictable large language model hallucinations into a scalable pipeline for building botnets out of developer machines and AI coding agents [1][2][3]. The technique extends the earlier "slopsquatting" pattern — attackers registering package names that AI models tend to invent — into repositories and agent "skill" files, and pairs that registration with prompt injection payloads that hijack an assistant's built-in terminal once the fabricated resource is fetched [1][2].

- HalluSquatting is untargeted and pull-based: attackers need no direct channel to a victim, because any agent that independently hallucinates the same resource name and retrieves it becomes compromised [3].
- Testing across widely used coding assistants and CLIs found hallucination rates as high as 85 percent for repository-cloning prompts and 100 percent for agent skill installations [2][3].
- The pattern is not theoretical: a hallucinated npm package named react-codeshift, first introduced through LLM-generated Agent Skill files, spread into 237 GitHub repositories before a researcher intervened [4].
- A parallel Palo Alto Networks study found roughly 250,000 hallucinated domains still unregistered across 913 analyzed brands, illustrating how far this class of exploitable hallucination extends beyond software packages [6].
- Effective mitigation likely requires layered defenses spanning AI application design, package registry governance, and enterprise controls on what agents are permitted to fetch and execute.

Background

The mechanics behind HalluSquatting are not new; they are the latest escalation of a pattern security researchers first documented under the name "slopsquatting" earlier in 2026. Large language models used for code generation periodically invent package names, repository identifiers, or API endpoints that sound plausible but do not exist, typically by blending the names of two legitimate, similarly purposed libraries. Because these fabrications are a function of how the model was trained rather than a random error, the same invented names recur across prompts, sessions, and even different model families.

Academic analysis of this behavior found 127 package names — 109 on PyPI and 18 on npm — that every major frontier model tested consistently hallucinated, with the researchers reporting that when identical prompts were repeated ten times each, 43 percent of the hallucinated names appeared in every single run [5]. That consistency plausibly explains why the underlying vulnerability is commercially exploitable: rather than guessing at random, an attacker could in principle run a modest batch of prompts, identify the names that recur most reliably, and pre-register them on public package registries seeded with malicious install scripts.

The real-world proof of this dynamic arrived in January 2026, when Aikido Security researcher Charlie Eriksen identified an npm package called react-codeshift that had never been legitimately published. The name was a blend of two real libraries, jscodeshift and react-codemod, and it first appeared embedded in a single commit containing 47 LLM-generated Agent Skill files — instructions that AI coding assistants read and execute when helping a developer complete a task. Nobody had deliberately planted the reference; an AI model had simply hallucinated it while generating the skill instructions, and the fabricated name then propagated organically as the repository containing it was forked and translated into other languages [4]. By the time Eriksen registered the name to study the phenomenon, it had already spread into 237 GitHub repositories, with automated agents attempting to install it on a daily basis [4]. A second documented case, a hallucinated package called unused-imports that models generate in place of the legitimate eslint-plugin-unused-imports, was still recording approximately 233 weekly downloads as of early February 2026 [4][5].

The scope of exploitable hallucination extends well beyond package registries. Palo Alto Networks' Unit 42 published research in July 2026 describing "phantom squatting," in which attackers register web domains that LLMs hallucinate when asked for a brand's official site, API endpoint, or developer portal. Analyzing 913 global brands across 685,339 queries against two frontier models, researchers generated 2.1 million candidate URLs, of which more than 13,229 were confirmed malicious, and identified roughly 250,000 additional hallucinated domains that remained unregistered and available for an attacker to claim [6]. In one documented instance, Unit 42 flagged a hallucinated postal-service domain 51 days before an attacker registered it, wrapped it in a convincingly cloned brand page, and used it to distribute a malicious Android application [6]. CSA's own research note on this phenomenon, published the same month, characterized it as evidence that hallucination-driven squatting has become a durable attacker technique rather than an isolated curiosity, one that spans domains, packages, and — as HalluSquatting now demonstrates — code repositories and agent skills [7].

Security Analysis

The HalluSquatting research, formally titled "Beware of Agentic Botnets: Scalable Untargeted Promptware Attacks via Universal and Transferable Adversarial HalluSquatting," was conducted by Aya Spira, Stav Cohen, Elad Feldman, Ron Bitton, Avishai Wool, and Ben Nassi, drawing on affiliations at Tel Aviv University, Technion, and Intuit [1][2][3]. The team's central contribution is demonstrating that hallucination-driven squatting generalizes beyond static package installation to any resource an agentic AI system might autonomously fetch and act upon, including GitHub repositories and agent skill definitions, and that compromising that fetch step can yield remote code execution rather than merely a poisoned dependency.

The attack chain begins the same way slopsquatting does: attackers profile which resource names a target model or family of models consistently hallucinates when asked to accomplish a coding task, then register those names in advance. Where HalluSquatting diverges is in what gets planted at the squatted location. Rather than a malicious install script tied to a single package manager, the attackers embed a resource — a repository or an agent skill file — that itself carries a prompt injection payload. When a developer asks a coding assistant such as Cursor, Cursor CLI, Windsurf, GitHub Copilot, Cline, Gemini CLI, or an OpenClaw-family assistant to clone a repository or install a skill, and the assistant hallucinates its way to the attacker's pre-registered resource, the embedded instructions hijack the assistant's own built-in terminal. Because many of these assistants are granted shell execution privileges as part of their normal operation, the injected instructions can install and launch botnet malware directly, without needing to exploit a separate software vulnerability [1][2][3].

The researchers classify this as an untargeted, pull-based indirect prompt injection attack, a distinction that matters for how defenders should think about exposure. Targeted prompt injection techniques — malicious content embedded in an email or calendar invite, for example — require the attacker to establish some delivery channel to a specific victim. HalluSquatting requires no such channel: because the hallucinated resource name is a predictable output of the model itself, any user or agent that independently arrives at the same hallucination and retrieves the resource becomes a victim, regardless of whether the attacker ever targeted that individual. This pull-based dynamic is what the researchers describe as making the attack "universal and transferable" — the same squatted resource can compromise many unrelated agents across many organizations without additional attacker effort per victim [3]. In testing, the team found hallucination rates reaching as high as 85 percent for prompts asking an agent to clone a repository, and 100 percent for prompts asking an agent to install a named skill [1][2][3].

The resulting compromise model is what the researchers term an "agentic botnet." As is well documented, traditional botnets such as Mirai depend on discovering vulnerable, poorly secured devices and exploiting known weaknesses or default credentials to gain a foothold. An agentic botnet instead aggregates developer workstations and enterprise systems that already run AI coding assistants with legitimate shell and network access, meaning the attacker inherits capability that would otherwise require lateral movement or privilege escalation to obtain. Once enrolled, compromised machines could in principle be directed toward the kinds of activity traditional botnets are used for — cryptocurrency mining, distributed denial-of-service campaigns, or staging for further intrusion — though the researchers' published materials describe the enrollment mechanism itself rather than documenting specific post-compromise campaigns [2][3]. This closely parallels a separate promptware technique CSA analyzed in March 2026, in which prompt injection payloads delivered via document analysis, email, and malicious websites enrolled multiple AI agents from different vendors into a unified command-and-control network capable of accepting natural language tasking [8]. HalluSquatting and that earlier promptware research describe two different delivery mechanisms — hallucination-driven pull versus content-based push — converging on the same underlying consequence: agent compromise that, once shell execution and network access are involved, is functionally indistinguishable from traditional host compromise [8].

Consistent with responsible disclosure practice, the research team notified affected vendors before publishing, and the published materials withhold specific exploit details the researchers judged could be directly reused by attackers [2]. That restraint limits near-term copy-paste exploitation, but in our assessment it does not reduce the underlying exposure: because the hallucination behaviors the attack depends on are intrinsic to how current frontier models generate code-related output, no single vendor is likely to patch this away without broader changes to how agentic tools verify what they fetch and execute.

Recommendations

Immediate Actions

- Treat every repository name, package reference, and skill identifier an AI coding assistant proposes as unverified until it is checked against an authoritative, known-good source, rather than assumed to exist because the model referenced it.
- Restrict AI coding assistants' and agents' shell execution and network egress privileges to the minimum required for their task, since HalluSquatting depends on the assistant's own terminal access to install malware.

- Require human confirmation before an agent clones a previously unseen repository or installs a new skill, particularly when that resource was suggested by the model itself rather than specified explicitly by the developer.
- Audit CI/CD pipelines and developer environments for evidence of hallucinated package names already in use, cross-referencing dependency manifests against known slopsquatting name lists as they become available from the security research community.
- Review incident response playbooks to account for compromise entering through an AI assistant's terminal rather than through a traditional network-facing vulnerability.

Short-Term Mitigations

Organizations should enforce lockfile pinning and cryptographic hash verification in build pipelines so that a dependency substitution — even one an AI assistant introduces unknowingly — cannot silently alter what code actually executes. Coding assistant configurations should require an explicit search or lookup step against an authoritative package or repository index before any fetch-and-install action proceeds, closing the gap that lets a model act directly on a hallucinated name. Security teams should also extend existing typosquatting and brand-monitoring programs to include hallucination-pattern monitoring, since the same registry-uniqueness and rapid-detection techniques used against traditional squatting apply directly to names an LLM is likely to invent. Finally, generating and reviewing a Software Bill of Materials for AI-assisted projects gives security teams a mechanism to catch a hallucinated dependency after the fact, even when it slipped past pre-installation checks.

Strategic Considerations

Over the longer term, the organizations building coding assistants and agent platforms are best positioned to close this gap structurally, by enforcing global uniqueness of repository and package names across registries, by preemptively registering the names their own models are most likely to hallucinate before an attacker can, and by strengthening malicious-content detection for newly registered repositories and skills. Enterprises evaluating AI coding tools should treat resistance to hallucination-driven supply chain attacks as a vendor selection criterion alongside more familiar security features, and should factor agentic tool exposure into their broader AI governance and Zero Trust programs rather than treating it as a narrow developer-tooling concern. As agentic coding assistants take on more autonomous responsibility for fetching and executing code, the distinction between a software supply chain incident and a network intrusion is likely to keep blurring, and governance frameworks should be updated accordingly.

CSA Resource Alignment

HalluSquatting sits at the intersection of two threat patterns CSA has already published dedicated research on this year, and organizations addressing it should treat those two notes as required companion reading. CSA's April 2026 research note, "Slopsquatting: AI Code Hallucinations Fuel Supply Chain Attacks," documents the foundational mechanism HalluSquatting builds on — attackers pre-registering package names that LLMs reliably hallucinate — and its recommendations on treating AI-generated package references as untrusted, enforcing lockfile pinning, and applying allowlists to autonomous agents apply directly to the repository- and skill-fetching behavior HalluSquatting exploits [11]. CSA's July 2026 note, "Phantom Squatting: AI Hallucinated Domains as Phishing Infrastructure," extends the same underlying insight to web domains, and its call for mandatory URL and resource verification before an agentic system fetches anything external maps closely onto the repository-cloning and skill-installation flows HalluSquatting targets [7].

The botnet consequence of successful compromise connects HalluSquatting to CSA's March 2026 research note, "Agent Commander: Promptware Turns AI Agents into C2 Infrastructure," which examined how prompt injection payloads can enroll compromised AI agents into a unified command-and-control network once they gain shell and network access [8]. That note's recommendations — auditing and revoking unnecessary agent capabilities, separating agent planning from execution contexts, and restricting network egress to approved destinations — describe exactly the controls that limit how far a HalluSquatting compromise can propagate after the initial hallucination-driven foothold is established. Both notes draw on CSA's MAESTRO framework for agentic AI threat modeling, which offers a layer-by-layer structure for reasoning about where in an agent's architecture a hallucination-driven compromise takes hold and what downstream layers it can reach [10].

For organizations building governance programs around these risks rather than point mitigations, the CSA AI Controls Matrix (AICM) v1.1 offers the broader control baseline. AICM v1.1's control objectives around supply chain and dependency validation are relevant to the pre-installation verification HalluSquatting defeats, and its model-security-related domains speak to the underlying hallucination behavior that makes the attack class possible. Organizations should reference AICM v1.1 when formalizing controls over what autonomous agents are permitted to fetch, install, and execute, rather than treating HalluSquatting as a one-off patching exercise [9].

References

[1] Ravie Lakshmanan. "New HalluSquatting Attack Could Trick AI Coding Assistants Into Installing Botnet Malware." The Hacker News, July 2026.

[2] SecurityWeek. "'HalluSquatting' Turns AI Hallucinations Into Botnet Delivery Mechanism." SecurityWeek, July 10, 2026.

[3] Aya Spira, Stav Cohen, Elad Feldman, Ron Bitton, Avishai Wool, Ben Nassi. "Beware of Agentic Botnets: Scalable Untargeted Promptware Attacks via Universal and Transferable Adversarial HalluSquatting." Tel Aviv University, Technion, and Intuit, 2026.

[4] Aikido Security. "Slopsquatting: The AI Package Hallucination Attack Already Happening." Aikido Security, 2026.

[5] arXiv. "The Range Shrinks, the Threat Remains: Re-evaluating LLM Package Hallucinations on the 2026 Frontier-Model Cohort." arXiv preprint, 2026.

[6] Unit 42, Palo Alto Networks. "Phantom Squatting: AI-Hallucinated Domains as a Software Supply Chain Vector." Palo Alto Networks, 2026.

[7] Cloud Security Alliance. "Phantom Squatting: AI Hallucinated Domains as Phishing Infrastructure." CSA AI Safety Initiative, July 2, 2026.

[8] Cloud Security Alliance. "Agent Commander: Promptware Turns AI Agents into C2 Infrastructure." CSA AI Safety Initiative, March 17, 2026.

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

[10] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA, February 2025.

[11] Cloud Security Alliance. "Slopsquatting: AI Code Hallucinations Fuel Supply Chain Attacks." CSA AI Safety Initiative, April 19, 2026.