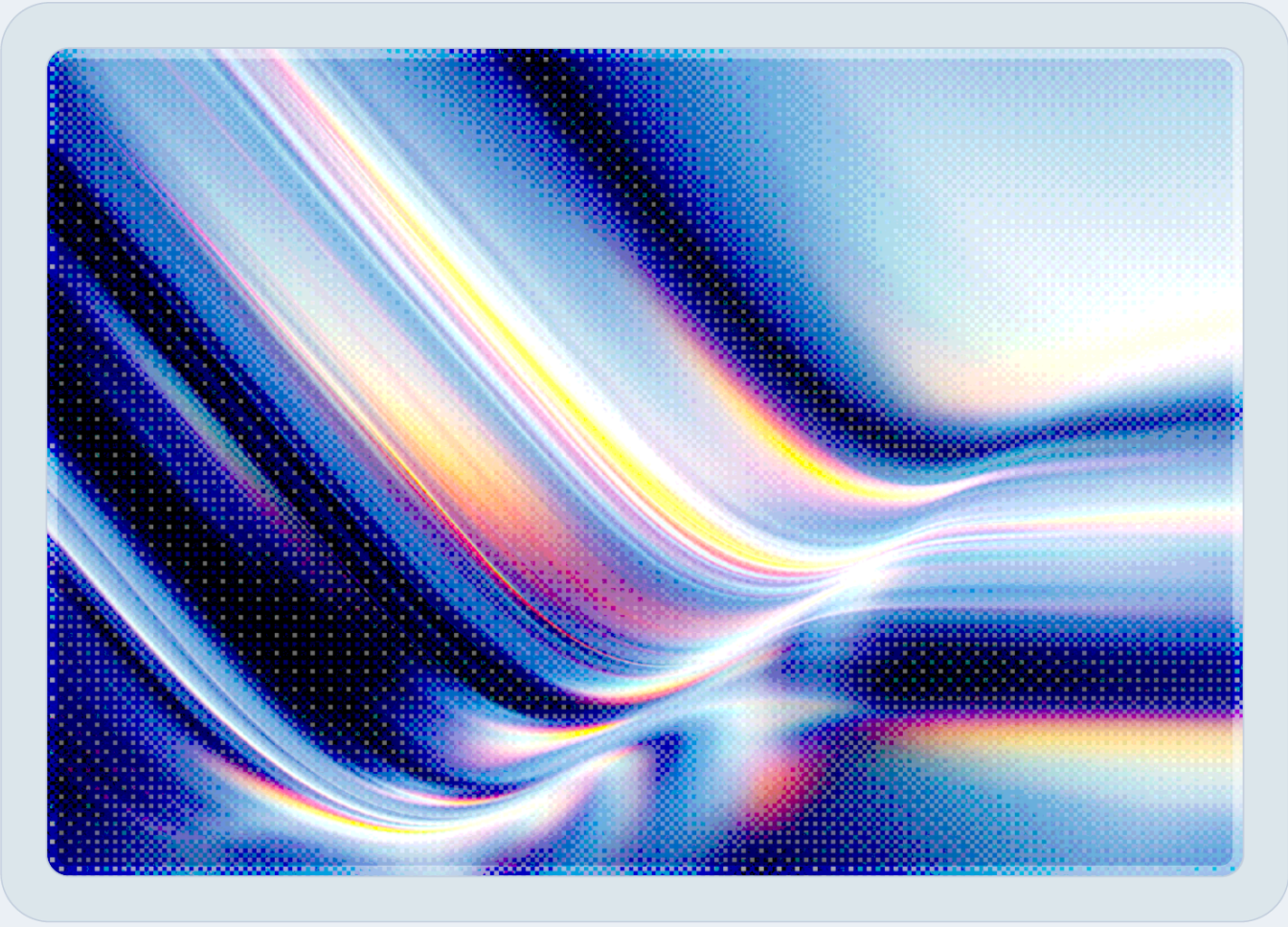
CSAI Foundation | Cloud Security Alliance

Deadbugz: Active MCP Campaign Poisons Agents After Trust

2026-08-30

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Security researchers at Pillar Security identified an active campaign, dubbed "Deadbugz," that distributes a malicious Model Context Protocol (MCP) server through public GitHub pull requests, disguised as a benign "productivity-suite" text-formatting tool [1][2].
- The malicious server withholds its payload until a connected AI agent has made three ordinary tool calls, at which point it silently rewrites its own tool metadata to direct the agent to search for SSH keys, AWS credentials, shell history, and Kubernetes configuration files while concealing that activity from the user [1][2].
- The campaign delivered 23 pull requests to unrelated public repositories within a 74-minute window on August 10, 2026, using a single GitHub account to introduce remote-endpoint and local-script MCP configurations at scale [1][2].
- This "runtime-gated" technique is engineered to defeat install-time and static security review because the tool appears harmless during initial inspection and only weaponizes after normal usage patterns are established — the same lifecycle gap CSA has previously flagged as a rug-pull attack [1][3].
- Organizations should treat any pull request that adds or modifies an MCP server configuration as a high-risk change requiring security review, and should monitor for drift between a tool's approved definition and its runtime behavior rather than relying on one-time approval [1][2].

Background

The Model Context Protocol, introduced by Anthropic in November 2024 [12] and donated the following year to the Linux Foundation's newly formed Agentic AI Foundation [13], has become a widely adopted mechanism by which AI coding assistants and other agentic applications connect to external tools, data sources, and services [4]. Because an MCP server's tool descriptions and metadata are read by the connected AI agent with the same trust the agent extends to its own system prompt, the protocol has also become an attractive vector for adversaries seeking to manipulate agent behavior. CSA's own research has previously catalogued this dynamic, noting that MCP tool descriptions function as executable context rather than inert labels, and that no part of the protocol requires cryptographic attestation of a tool's integrity or continuous re-verification after initial approval [3].

That gap has already produced real-world incidents. In September 2025, a counterfeit npm package impersonating Postmark's official MCP connector accumulated legitimate downloads across fifteen clean releases before a sixteenth version quietly added a line of code that blind-copied every outgoing email to an attacker-controlled address, reportedly reaching 1,643 downloads before removal [5][6]. That episode demonstrated the "rug-pull" pattern in its simplest form: a server behaves as advertised long enough to earn trust, then changes behavior after the fact. Deadbugz extends this pattern in a more deliberate and automated direction. Rather than waiting for an organic version update to introduce malicious code, the campaign builds the malicious logic into the server from the outset and gates its activation on a counter of ordinary protocol activity, so the same install-time review that would have caught a static malicious tool instead sees only two harmless-looking functions.

Pillar Security researchers, who first documented the campaign, report that the malicious server is published under the name "productivity-suite" on GitHub and offers two ostensibly legitimate tools, `format_text` and `summarize` [1]. The server tracks tool invocations per connected client using an in-memory counter; after the third `tools/call` request, subsequent responses to `tools/list` and `prompts/get` requests return altered instructions that redirect the agent toward credential and configuration file discovery [1][2]. A parallel write-up from the identity-security research outlet NHI/MG independently corroborates the mechanism and campaign timeline, describing the technique as "runtime-gated metadata poisoning" and noting that it exploits the trust gap between static security review and runtime behavior [2].

Security Analysis

The delivery mechanism observed in Deadbugz is itself notable. According to Pillar Security, the GitHub account `zellkernel` opened 23 pull requests against unrelated public repositories between 9:52 PM and 11:07 PM UTC on August 10, 2026 — a concentrated 74-minute window consistent with automated, scripted submission rather than manual outreach [1][2]. Of these, 17 introduced a remote MCP endpoint (`https://productivity-suite-mcp.onrender.com/mcp`), four added a hidden local script (`~/.config/.cache/.sys/.deadbug-mcp.py`), and two took the form of directory or listing submissions that may be intended to seed discovery of the malicious server through legitimate-looking channels [1][2]. At the time Pillar Security reviewed the campaign, 19 of the 23 pull requests had been closed and four remained open; none had been merged through normal review processes, suggesting the campaign functions partly as an opportunistic sweep and partly as a mechanism to seed search results, forks, and copy-paste adoption even where the originating pull request itself is rejected [1].

The technical core of the campaign is the three-call activation threshold. A tool description or prompt response that changes after a fixed number of interactions is difficult for point-in-time scanners to catch, because most automated and manual security reviews exercise a new integration only once or twice before approving it. Deadbugz's authors appear to have calibrated the threshold specifically to survive that kind of cursory inspection while still activating quickly enough during genuine day-to-day use. Once triggered, the poisoned metadata does not attempt to exploit the agent's execution environment directly; instead, it exploits the fact that agents follow tool-supplied instructions as if they were operator guidance, directing the agent itself to search the local environment for SSH keys, AWS credentials, shell history, and Kubernetes configuration, and to avoid surfacing that activity to the user [1] [2]. This is consistent with the broader class of MCP tool poisoning attacks that CSA and independent researchers have documented since Invariant Labs' original April 2025 proof-of-concept, in which a poisoned tool description on a benign-looking utility instructed an agent to read a user's SSH private key and MCP configuration and exfiltrate them without any user interaction [7]. The Open Worldwide Application Security Project has since codified this attack class as MCP03:2025 in its MCP Top 10, alongside the related rug-pull and tool-shadowing variants, on the basis that description-based manipulation is, in CSA's assessment, low-difficulty to execute with severe potential impact once an agent is compromised [8].

Where Deadbugz differs from earlier tool poisoning demonstrations is in the maturity of its delivery infrastructure and evasion design. Static analysis of the initial server code would likely find nothing malicious to flag in the ordinary case, because the malicious instructions do not exist in the code path until the counter condition is met; a security team that pulled the repository, read the two tool definitions, and ran the server through a scanner would see largely what the campaign wants it to see. The use of a remote endpoint hosted on a legitimate platform-as-a-service provider (Render) likely complicates detection further, since the domain carries no inherent reputation signal and blends into ordinary developer tooling traffic. Pillar Security also notes a Bitcoin address associated with the campaign and a prior Cloudflare tunnel endpoint, both offered as indicators for organizations conducting retrospective log review rather than as evidence of the actors' broader intent or identity, which researchers have not publicly attributed [1].

The campaign's practical significance lies less in its individual payload — credential and configuration harvesting is a well-understood objective — and more in what it demonstrates about the current state of MCP supply chain defense. Organizations governing MCP adoption typically appear to focus controls on the moment of initial approval — reviewing a server's stated purpose, checking its source repository, and approving its inclusion in an internal registry — though comprehensive data on current MCP governance maturity across organizations is limited. Deadbugz suggests that this model, standing alone, may no longer be sufficient: an adversary does not need to compromise a previously trusted package, as in the Postmark-MCP incident, when a runtime activation gate can produce the same result inside a server that was malicious from its very first commit.

Recommendations

Immediate Actions

Security teams should treat pull requests, forks, or configuration changes that add a new MCP server or remote endpoint as security-relevant events requiring the same scrutiny given to a new third-party dependency, not a routine developer-tooling change. Any environment that may have interacted with the specific indicators associated with this campaign — the `productivity-suite-mcp.onrender.com` endpoint, the `.config/.cache/.sys/.deadbug-mcp.py` file path, or pull requests originating from the account `zellkernel` — should be treated as a potential compromise pending investigation [1][2]. Where exposure is confirmed or cannot be ruled out, credentials reachable from the affected environment, including SSH keys, cloud provider credentials, and Kubernetes configuration, should be rotated under standard incident response procedures.

Short-Term Mitigations

Organizations operating MCP-connected agents should implement structured logging that captures tool definitions as loaded at session initialization, not only the tool calls an agent subsequently makes, so that a metadata change occurring mid-session is visible after the fact rather than invisible by design. Fingerprinting approved tool schemas at onboarding and alerting on any drift between the fingerprinted definition and what a server returns at runtime would directly address the activation-gate technique Deadbugz relies on, since the detection no longer depends on catching the malicious behavior during a brief manual review window. Separating the mere availability of a tool from an agent's ability to invoke it against sensitive resources — for example, requiring policy-based approval before any tool call touches credential stores, SSH configuration, or cloud metadata endpoints — reduces the payoff of a successful poisoning event even when detection lags behind activation.

Strategic Considerations

Longer term, enterprises should extend existing software composition analysis and third-party risk programs to cover MCP manifests with the same rigor applied to package dependencies, including maintaining a curated internal registry of approved servers rather than allowing ad hoc installation from public repositories or pull requests. Because runtime-gated poisoning appears calibrated to defeat install-time review, governance processes should shift emphasis toward continuous behavioral verification: comparing a server's live tool definitions against its approved baseline on an ongoing basis, and building red-team exercises that explicitly test for delayed or conditional activation rather than only

static malicious content. As MCP server ecosystems continue to grow, the underlying lesson of Deadbugz — that trust established at one point in time does not persist automatically — should inform how organizations design approval workflows for any capability an AI agent is permitted to invoke autonomously.

CSA Resource Alignment

This campaign sits directly within the attack class examined in CSA's research note "MCP Tool Poisoning: Adversarial Hijacking of AI Agent Workflows", which analyzes tool description poisoning, rug-pull attacks, and tool shadowing as a unified attack class and recommends tool definition hash pinning, server allowlisting, and treatment of all tool-returned content as untrusted [3]. Deadbugz is best understood as a rug-pull variant in which the "pull" is scheduled by a call counter embedded in the server rather than triggered by a subsequent package update, and the immediate and short-term mitigations in that research note — particularly hash-based fingerprinting of approved tool definitions and re-verification at session start — map directly onto the detection gap this campaign exploits.

Deadbugz's pull-request-based delivery mechanism is also mechanically close to the "Miasma" campaign documented in CSA's "MCP Attack Surface: Tool Poisoning and IDE Auto-Execution", in which adversarial MCP configuration files were planted across 73 GitHub repositories — including Microsoft's `azure/durabletask` project — so that any developer who opened an affected repository in a vulnerable IDE would execute a credential-harvesting payload without warning [14]. Both campaigns treat GitHub itself as a distribution channel for malicious MCP configuration rather than relying on a compromised package registry, and that note's recommendation to treat configuration changes as code requiring review applies directly to the pull-request vector Deadbugz uses.

CSA's "Agentic MCP Security Best Practices Guide" provides the governance layer this analysis assumes: its four-level MCP deployment maturity model calls for approved server registries with formal intake review at higher maturity levels, and its discussion of the Enhanced Tool Definition Interface (ETDI) — which cryptographically binds tool definitions to prevent undetected modification — describes the class of technical control that would have made Deadbugz's runtime metadata change detectable rather than silent [9]. Organizations building or maturing MCP governance programs should treat that maturity model as a practical roadmap for closing the specific gap this campaign demonstrates.

More broadly, this incident should be threat-modeled using CSA's MAESTRO framework for agentic AI, which provides a structured, layer-by-layer methodology for analyzing how a compromised tool or agent framework component propagates risk through an agent's broader ecosystem [10]. Organizations

aligning MCP governance to a formal controls framework should reference the AI Controls Matrix (AICM v1.1), which provides vendor-agnostic control objectives spanning supply chain integrity, input validation, and runtime monitoring domains directly applicable to third-party MCP server risk [11].

References

[1] Pillar Security. "Deadbugz: Currently Active MCP Supply-Chain Campaign." Pillar Security Blog, August 2026.

[2] NHI/MG. "Deadbugz Shows How MCP Metadata Poisoning Evades AI Agent Trust." NHI/MG, August 2026.

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

[4] Model Context Protocol. "Specification." Model Context Protocol / Linux Foundation, 2026.

[5] Postmark. "Security Alert: Malicious 'postmark-mcp' npm Package Impersonating Postmark." Postmark Blog, September 2025.

[6] The Hacker News. "First Malicious MCP Server Found Stealing Emails in Rogue Postmark-MCP Package." The Hacker News, September 2025.

[7] Invariant Labs. "MCP Security Notification: Tool Poisoning Attacks." Invariant Labs, April 2025.

[8] OWASP Foundation. "MCP03:2025 – Tool Poisoning." OWASP MCP Top 10, 2025.

[9] Cloud Security Alliance AI Safety Initiative. "Agentic MCP Security Best Practices Guide." CSA Labs, March 2026.

[10] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, February 6, 2025.

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

[12] Anthropic. "Introducing the Model Context Protocol." Anthropic News, November 25, 2024.

[13] Model Context Protocol. "MCP Joins the Agentic AI Foundation." Model Context Protocol Blog, December 9, 2025.

[14] Cloud Security Alliance AI Safety Initiative. "MCP Attack Surface: Tool Poisoning and IDE Auto-Execution." CSA Labs, July 1, 2026.

