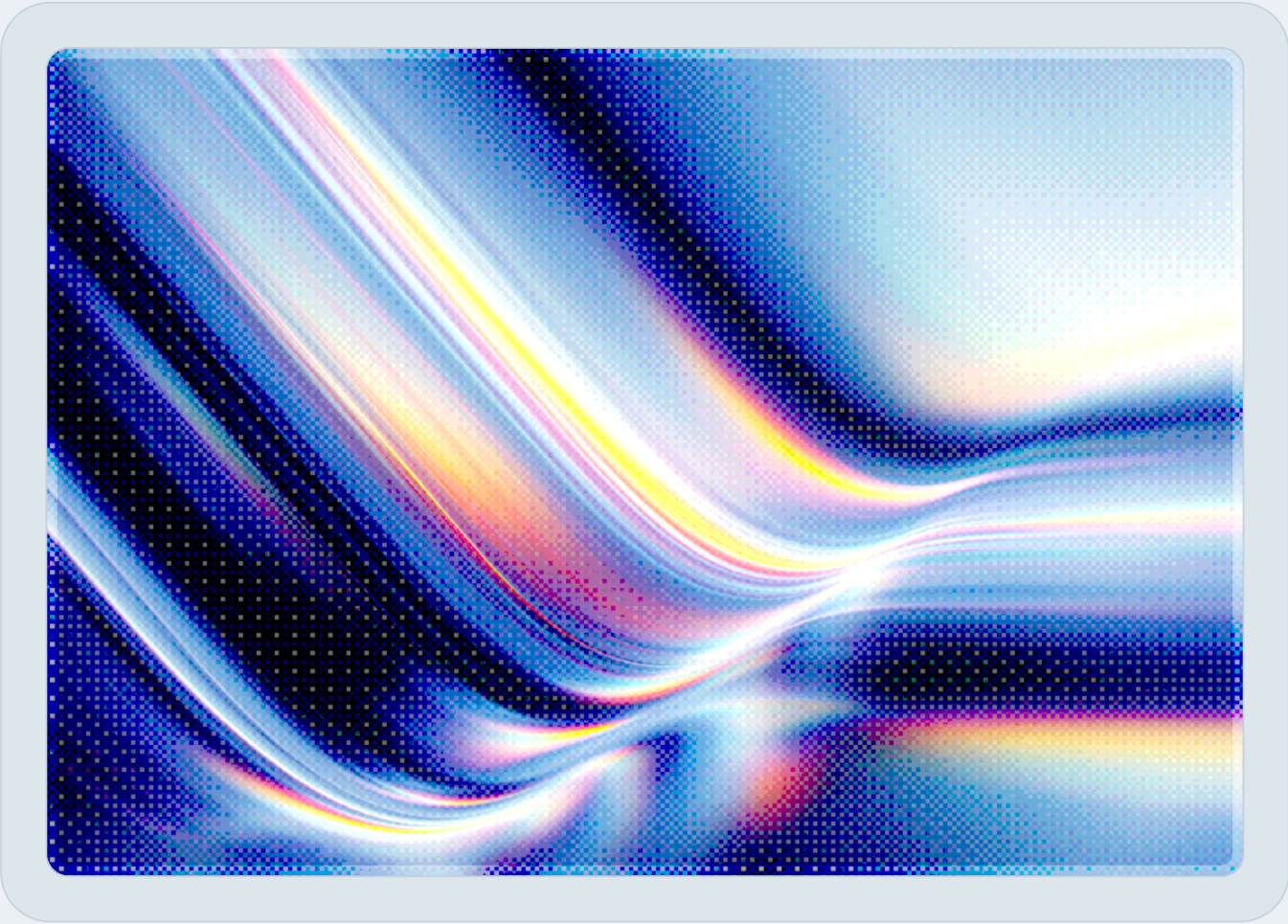
CSAI Foundation | Cloud Security Alliance

AutoJack and Agentjacking: AI Agent Frameworks as a New RCE Attack Surface

2026-06-23

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- **AutoJack** (disclosed June 18, 2026) demonstrates that a single malicious webpage can achieve remote code execution on any host running affected pre-release versions of AutoGen Studio's browsing agent (0.4.3.dev1 or 0.4.3.dev2), exploiting a chain of three vulnerabilities in the framework's authentication and parameter handling design [1].
- **Agentjacking** (disclosed by Tenet Security, June 2026) shows that malicious content injected into Sentry error-tracking data can hijack AI coding agents, with researchers identifying over 2,388 potentially exposed organizations [2][3].
- Both attacks illustrate a recurring failure across surveyed agent frameworks: trust models that assume their execution environment is safe, rather than treating every input source — including localhost services and monitoring tools — as a potential adversary.
- Traditional security controls — endpoint detection, web application firewalls, identity and access management, and VPN perimeters — are not reliably positioned to detect either attack class with default configurations, because each individual action in the exploit chain is individually authorized by the framework [2].
- Organizations deploying agentic AI must apply infrastructure-grade security controls to agent execution environments, including explicit authentication for all agent control planes and strict scoping of tool permissions.

Background

The AI agent ecosystem has expanded rapidly over the past eighteen months. Where organizations once deployed large language models through simple API integrations, they now run autonomous agent frameworks — software systems that give language models persistent memory, access to external tools, and the ability to execute code, browse the web, and interact with cloud services on behalf of users. Frameworks such as AutoGen, LangChain, CrewAI, and dozens of others have moved from research prototypes to production deployments across enterprise environments, software development pipelines, and, increasingly, security operations tools.

This architectural shift has fundamentally changed the threat surface. A language model accessed through a stateless API call poses a qualitatively different risk than an agent framework that maintains a persistent local process, exposes control-plane endpoints on localhost, connects to monitoring services and version control systems, and executes arbitrary tool calls in response to model output. The security community's prior characterization of prompt injection as primarily a content moderation concern has been superseded by the architectural evolution of agent frameworks: as agents gained direct tool execution capabilities, prompt injection became a remote code execution primitive.

The Model Context Protocol (MCP), introduced by Anthropic in late 2024 and now embedded across most major agent frameworks, has amplified this dynamic. MCP standardizes how agents connect to external tools — file systems, databases, web browsers, monitoring platforms, and development environments — but its adoption outpaced security design. The protocol's initial implementations include a structural architecture in which any process command passed to the MCP STDIO interface executes on the host system, regardless of validation [4]. As MCP adoption expanded rapidly across framework ecosystems in 2025, with thousands of publicly accessible servers catalogued by security researchers [11], it carried this exposure into a wide range of production deployments.

AutoJack and Agentjacking represent the maturation of this threat landscape: proof-of-concept research has given way to concrete, reproducible attacks against specific framework versions in widespread use.

Security Analysis

AutoJack: Localhost Trust Enables Single-Page RCE

The AutoJack vulnerability, disclosed by Microsoft's Defender Security Research Team on June 18, 2026, targets AutoGen Studio, Microsoft's browser-based interface for building and running multi-agent workflows [1]. The attack chain requires no credentials, no social engineering beyond causing a user to visit a website, and no elevated privileges on the target machine. When a developer running AutoGen Studio visits a malicious webpage with their browser, that page can silently execute arbitrary commands on the developer's host system with the privileges of the AutoGen Studio process.

The exploit combines three separate weaknesses in the framework's architecture. The first is a structural limitation of localhost trust assumptions: browsing agents configured within AutoGen Studio issue HTTP requests on behalf of the local application, meaning that when such an agent loads a malicious page,

that page appears to originate from localhost and is permitted to make requests to other localhost services. This is not an access control bypass in the traditional sense; it reflects a design assumption that the localhost boundary provides meaningful protection when it does not.

The second weakness is the absence of authentication on AutoGen Studio's MCP WebSocket endpoint. The framework's middleware was configured to exclude routes under `/api/mcp/*` from standard authentication checks, leaving the agent's control plane accessible to any process able to reach it — including the malicious webpage now executing inside the browsing agent's context.

The third weakness concerns parameter handling: the MCP WebSocket endpoint accepts base64-encoded `server_params` values supplied directly in URL query parameters and passes them, unvalidated, to the code responsible for launching external processes. An attacker who can reach this endpoint — which, by the first two steps, is any webpage the browsing agent visits — can supply arbitrary command strings and have them executed on the host. Together, the three weaknesses form a kill chain that transforms a developer's ordinary web browsing activity into a code execution opportunity for any operator of a malicious website.

Microsoft has issued a fix in the AutoGen Studio GitHub repository (commit b047730, PR #7362), but as of this writing the patch has not been incorporated into a stable PyPI release. Pre-release versions 0.4.3.dev1 and 0.4.3.dev2 remain available on PyPI and are affected. The stable 0.4.2.2 release, which lacks the MCP route entirely, is unaffected [1][5].

Agentjacking: MCP Tool Poisoning via Monitoring Infrastructure

Where AutoJack exploits a misconfigured control plane, agentjacking exploits the trust that AI coding agents place in the outputs of their integrated tools. The attack class, named by Tenet Security in their June 2026 research [2], is illustrated concretely through Sentry, the widely used open-source error-tracking platform.

Sentry's MCP server connects AI coding agents — including Claude Code, Cursor, and OpenAI Codex — to a developer's error tracking data, enabling agents to inspect crash reports and propose fixes. The integration is appropriate for its stated purpose of surfacing error data to agent context; the vulnerability lies not in the integration's design but in the agent's inability to distinguish legitimate remediation guidance from attacker-controlled instructions embedded in the same data structure. Agentjacking exploits precisely this trust relationship. Because Sentry's event ingestion accepts arbitrary payloads from any party who possesses the project's public DSN key, an adversary who can inject content into Sentry events can supply instructions that the AI coding agent will receive as part of its trusted tool context.

Tenet's research found that 2,388 organizations could be identified as potentially exposed, and that in controlled testing environments, AI coding agents executed injected payloads with an 85% success rate [2][3]. The demonstrated impact includes exfiltration of environment variables, AWS credentials, GitHub tokens, and private repository URLs. In CI/CD pipeline configurations, such credential access would create pathways to broader infrastructure compromise — including repository write access, artifact signing, and cloud resource manipulation — though this downstream impact was not demonstrated in Tenet's controlled testing [14]. Tenet Security has committed to open-sourcing defensive tooling called Agent-JackStop, providing configuration hardening for Cursor and Claude Code environments, though this has not been released as of this writing.

The reason agentjacking is particularly difficult to defend against using conventional security tools is that no individual action in the attack chain is unauthorized. The Sentry integration is a legitimate, configured tool. The agent reading error event data is performing its expected function. The subsequent shell commands or file operations that execute the injected payload are issued through the framework's normal tool invocation mechanisms. Only the sequence as a whole constitutes an attack, and that sequence is invisible to endpoint detection tools observing individual process calls [2].

Convergent Failure Modes in the Broader Ecosystem

AutoJack and agentjacking are not isolated incidents but representative examples of a broader pattern documented across the agentic AI framework ecosystem in 2025 and 2026. Microsoft's security research team, following a year of red-teaming agentic systems, published a revised failure mode taxonomy identifying seven new vulnerability categories: agentic supply chain compromise, goal hijacking, inter-agent trust escalation, computer use agent visual attacks, session context contamination, MCP and plugin abuse, and capability disclosure [6]. That body of research also demonstrated concrete zero-click bypasses of human-in-the-loop controls and tool parameter injection paths leading to remote code execution [6][13].

CVE-2025-68664 in LangChain, assigned a CVSS score of 9.3, describes a deserialization vulnerability in which user-controlled data is treated as a legitimate LangChain object during serialization operations, enabling exfiltration of API keys and secrets from affected deployments [7]. CrewAI's sandbox implementation, catalogued under CERT Coordination Center vulnerability note VU#221883, defaults to an execution mode that fails to block `ctypes` calls when Docker is unavailable, permitting arbitrary system command execution through standard Python library calls [8]. CVE-2025-6514, a critical 9.6 CVSS vulnerability in the `mcp-remote` package, enables arbitrary OS command execution when a host connects to an untrusted MCP server [9]. Each represents the same pattern observed across surveyed frameworks: execution environments that lack the defense-in-depth expected of other network-accessible software.

CSA's prior research note on agentjacking, published June 12, 2026, addressed the Sentry MCP vector specifically [10]. AutoJack extends the threat model to encompass the browsing agent attack surface and reinforces that the shared root cause across both attacks is the absence of architectural trust boundaries in agentic framework design.

Recommendations

Immediate Actions

Organizations running AutoGen Studio should audit which version is deployed. The affected pre-release versions (0.4.3.dev1 and 0.4.3.dev2) should be removed or replaced with the stable 0.4.2.2 release until a patched stable release incorporating commit b047730 is available. Development environments in which AutoGen Studio's browsing agent is active should not be used for general web browsing on the same host or browser session. Developers using AI coding agents integrated with Sentry should audit their Sentry DSN key exposure: if the DSN is accessible in public repositories, CI configuration files, or client-side code, it can be used to inject malicious error events without access to the Sentry account itself.

Short-Term Mitigations

Agent control planes — including MCP WebSocket endpoints, AutoGen Studio's local API, and analogous local HTTP services in other frameworks — must be treated as privileged network services rather than trusted internal interfaces. This means applying the same authentication requirements to localhost-bound endpoints that would apply to externally exposed APIs, including session-bound tokens that cannot be forged by content the agent loads during task execution.

Tool scoping is a second critical control. The set of tools available to an agent should be scoped to the minimum necessary for the task being performed, and tools that expose high-impact capabilities — shell execution, credential access, repository write operations — should be disabled or require explicit approval for each invocation. The fact that agentjacking is effective in part because agents with Sentry integration also have access to shell execution tools illustrates this: separating monitoring access from execution access would have disrupted the demonstrated kill chain, removing the agent's ability to act on injected payloads. OWASP has documented MCP tool poisoning as a formal attack category [15], providing a structured taxonomy that security teams can use to assess their control coverage against this threat class.

Organizations should validate their MCP server configurations against the known vulnerability database maintained at vulnerablemcp.info [11], and prioritize updating any `mcp-remote` deployments to versions that have addressed CVE-2025-6514 [9].

Strategic Considerations

The agentic AI attack surface requires a procurement and deployment posture that does not yet exist in most organizations. AI agent frameworks should be evaluated against security criteria analogous to those applied to web application frameworks — including authenticated control planes, parameterized tool invocation, sandboxed execution of agent output, and defined trust boundaries between agent sessions. Organizations building internal agent tooling should apply the same principles by design rather than by remediation.

Security teams should extend their threat modeling to encompass agent execution environments specifically. The MAESTRO threat taxonomy published by CSA [18] provides a framework for categorizing these risks, including Tool Injection (Layer 5) and Agent Hijacking (Layer 3) scenarios that directly map to the AutoJack and agentjacking patterns documented here. Red-teaming programs should explicitly exercise indirect prompt injection vectors — content that enters the agent context through tools, monitoring integrations, email, calendar data, or document retrieval — rather than focusing only on direct user-interface injection [16].

CSA Resource Alignment

The vulnerability classes documented in this note map to several active CSA research and framework initiatives. CSA's Agentic AI Red Teaming Guide provides testing methodology specifically designed for identifying tool injection and agent hijacking vulnerabilities of the type exploited in both attacks, and organizations should apply this guidance to their agentic AI deployments prior to production launch [12]. The MAESTRO threat modeling framework [18] addresses both the Tool Injection scenario (Layer 5: Tools and Extensions) and Agent Hijacking (Layer 3: Agent Trust and Communication) as first-class threat categories, giving security teams a structured vocabulary for risk assessment and control selection.

The AI Controls Matrix (AICM) v1.0 [17] addresses several relevant control domains. The Application Provider (AP) domain includes least-privilege enforcement for tool access and input validation requirements that apply directly to MCP integration design. The Supply Chain (SC) domain governs vetting of third-party tools and plugins that enter the agent context — the vector exploited in

marketplace poisoning attacks that preceded AutoJack and agentjacking as early warning indicators. The AI Security Testing (AS) domain specifies continuous evaluation requirements that should now include indirect prompt injection and agent hijacking test cases as mandatory coverage areas.

CSA's Agentic Trust Framework (ATF), stewarded in partnership with MassiveScale.AI, provides governance structures for autonomous agent systems, including accountability chains, oversight mechanisms, and criteria for human-in-the-loop intervention. The findings here underscore the ATF's core argument: agentic AI systems cannot be governed through the same controls applied to deterministic software, because their attack surface includes not only their inputs and outputs but the full context they consume during task execution.

The research note on agentjacking published by CSA Labs on June 12, 2026 [10] and this note together establish a documented pattern of MCP-mediated and control-plane attacks that the broader CSA community should treat as an ongoing class, not an isolated pair of incidents. Organizations are encouraged to review both notes in conjunction with the MAESTRO threat taxonomy to assess their exposure across the full agentic attack surface.

References

[1] Microsoft Defender Security Research Team. "AutoJack: How a single page can RCE the host running your AI agent." Microsoft Security Blog, June 18, 2026.

[2] Tenet Security. "One Fake Bug Report Hijacked a $250B Company's AI Agent." Tenet Security Blog, June 2026.

[3] THN Staff. "Agentjacking Attack Tricks AI Coding Agents Into Running Malicious Code." The Hacker News, June 2026.

[4] CSA Labs. "MCP by Design: RCE Across the AI Agent Ecosystem." Cloud Security Alliance, April 2026.

[5] S. Gatlan. "Microsoft fixes AutoGen Studio flaw that enabled code execution." BleepingComputer, June 23, 2026.

[6] Microsoft Security Response Center. "Updating the taxonomy of failure modes in agentic AI systems." Microsoft Security Blog, June 4, 2026.

[7] Miggo Research. "CVE-2025-68664: LangChain Deserialization Vulnerability." Miggo, 2025.

[8] CERT/CC. "VU#221883 - CrewAI contains multiple vulnerabilities." Carnegie Mellon University CERT Coordination Center, 2026.

[9] JFrog Security Research. "CVE-2025-6514: Critical RCE Vulnerability in mcp-remote." JFrog, 2025.

[10] CSA Labs. "Agentjacking: MCP Injection Hijacks AI Coding Agents." Cloud Security Alliance, June 12, 2026.

[11] vulnerablemcp.info. "The Vulnerable MCP Project: Comprehensive Model Context Protocol Security Database." Accessed June 2026.

[12] CSA AI Organizational Responsibilities Working Group. "Agentic AI Red Teaming Guide." Cloud Security Alliance, 2025.

[13] Microsoft Security Blog. "When prompts become shells: RCE vulnerabilities in AI agent frameworks." Microsoft, May 7, 2026.

[14] Microsoft Security Blog. "Securing CI/CD in an agentic world: The Claude Code GitHub Action case study." Microsoft, June 5, 2026.

[15] OWASP Foundation. "MCP Tool Poisoning." OWASP, 2026.

[16] NIST. "Technical Blog: Strengthening AI Agent Hijacking Evaluations." National Institute of Standards and Technology, January 2025.

[17] Cloud Security Alliance. "AI Controls Matrix (AICM)." CSA, 2025.

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