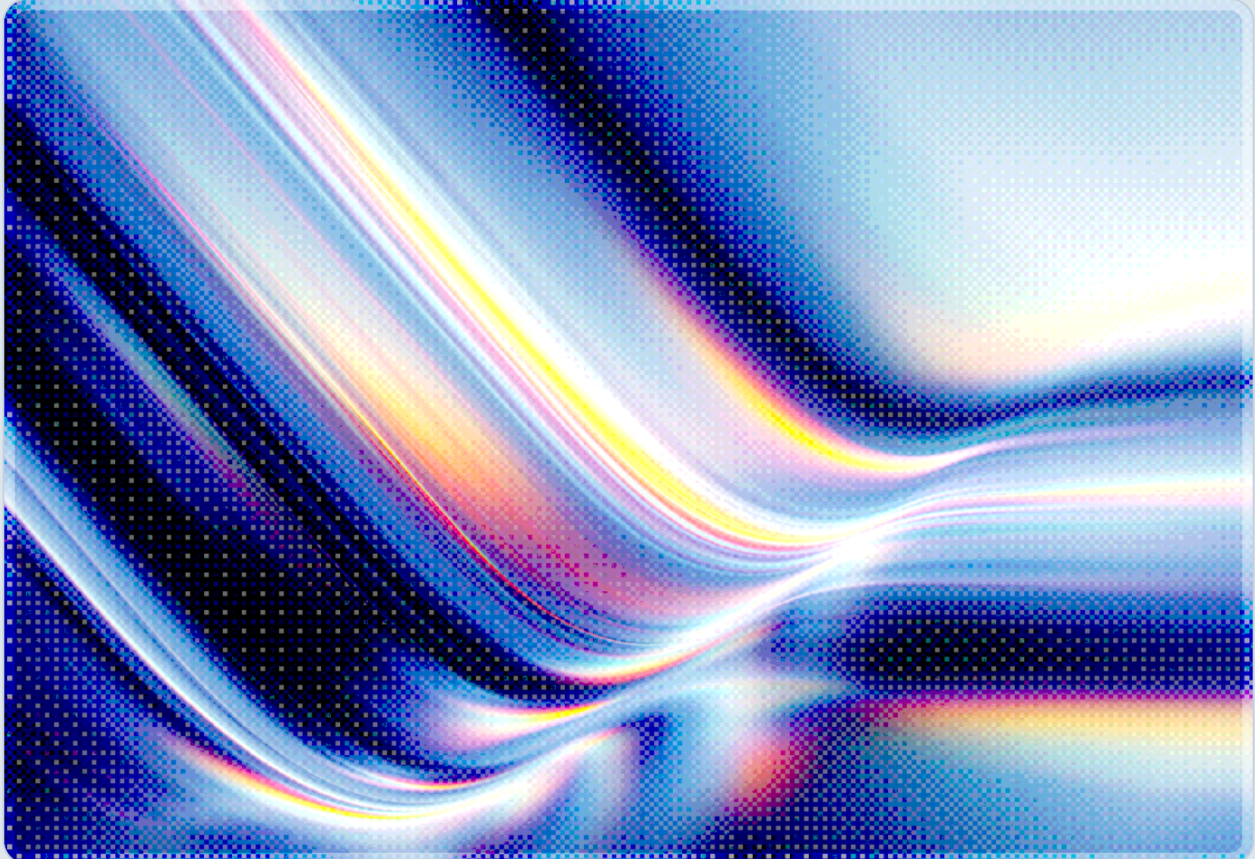


MCP Launch Paths as RCE: The opencode OAuth Flaw and Copilot CVE-2026-103572

Security Implications and Guidance

2026-10-22

 AI-assisted Rapid Research



CSAI

cloud
security
alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

Two disclosures in October 2026 show the same weakness in two different AI coding tools. In each case, data an attacker could influence reached the code that launches operating system processes for Model Context Protocol (MCP) integrations. In opencode, the open-source coding agent maintained by Anomaly, a remote MCP server could plant a PowerShell subexpression in the OAuth authorization endpoint it advertised. On Windows and WSL, that string ran as a command when the client tried to open a browser for sign-in [1][2]. In GitHub Copilot for Visual Studio and SQL Server Management Studio (SSMS), tracked as CVE-2026-103572, a prompt-injected agent could append a trailing dot to a filename to get past the protection on `.mcp.json`. It could then register an attacker-chosen command as an MCP server, which the IDE would launch [3]. The opencode flaw has no CVE; the CVE identifier in the title applies only to the Copilot issue.

The first priority is patching. Organizations should upgrade opencode to 1.18.33 or later and apply the October 2026 Visual Studio security updates, and SSMS users should confirm with Microsoft which update applies to their build [1][3][8]. Because no CVE or vendor advisory exists for the opencode flaw, vulnerability scanners and dependency bots are unlikely to flag vulnerable installations as of this writing, so security teams should inventory them directly [1][2].

Beyond patching, the two cases argue for treating MCP launch configuration as executable code. Any file or field that determines what an MCP client spawns, whether a stdio server command or a URL passed to a browser launcher, should be treated as a code-execution primitive. Validation that inspects a different representation from the one finally executed does not stop these attacks. The opencode payload sat inside a valid `https:` URL [1], and the Copilot bypass passed a string-comparison check that ran before the path was canonicalized [3].

Finally, vendor threat models differ on server trust. Anomaly told the researchers that malicious MCP servers fall outside its threat model [1][4]. Organizations should therefore make their own decisions about which MCP servers to trust and should not assume the client provides that boundary.

Background

The Model Context Protocol lets an AI client connect to external tools and data sources, which MCP calls "servers." A local server is usually a stdio subprocess: the client reads a configuration file that names an executable and its arguments, then launches it with the user's privileges. A remote server is reached over HTTP. If a remote server requires authentication, it returns a `401` challenge, and the client follows OAuth discovery metadata to find the authorization endpoint before opening the user's browser to sign in [1]. Each pattern includes a moment where the client converts data into a process launch. This note calls that moment the "launch path." For stdio servers, the configuration file is the launch path; for remote OAuth servers, the browser-launch helper is.

This vulnerability class is not new. In July 2025, JFrog disclosed CVE-2025-6514 in `mcp-remote`, a widely used proxy that connects local MCP clients to remote servers. A malicious server could supply a crafted `authorization_endpoint`, which reached an `open()` call. On Windows this gave arbitrary command execution, which Pillar Security characterizes as relying on PowerShell subexpression evaluation. The issue was rated CVSS 9.6 and fixed in version 0.1.16 [1][5][6]. Pillar's opencode research connects the new finding to that CVE and to other advisories involving the same sink [1]. The opencode disclosure shows that the lesson from CVE-2025-6514 did not reach at least one major MCP client, and the Copilot case shows that the configuration-file variant of the launch path can be manipulated by the agent itself.

opencode is a terminal-based, open-source AI coding agent whose repository had between roughly 211,000 and 218,000 GitHub stars at the time of disclosure, according to Pillar and Infosecurity Magazine respectively [1][2]. GitHub Copilot in Visual Studio and SSMS is Microsoft's agentic coding assistant, built into the IDE shell that both products share [3]. The opencode disclosure came out within days of a separate, unrelated breach of opencode's hosted Console that may have exposed stored provider API keys [2][4]. This note covers only the code-execution flaw.

Security Analysis

opencode: OAuth Discovery Becomes a Shell Command

Pillar Security researcher Eilon Cohen traced a data flow from a remote MCP server's OAuth discovery document through the MCP SDK and opencode's OAuth provider, ending in a browser-launch helper [1]. opencode had pinned version 10.1.2 of the `open` npm package. On Windows, that version builds a PowerShell command, base64-encodes it, and runs it with `powershell -EncodedCommand`. The

application name is escaped, but the target URL is not, and it is wrapped in double quotes. PowerShell evaluates `$(...)` subexpressions inside double-quoted strings, so a URL such as `https://evil.test/$(...)` runs the embedded expression before the browser launch fails. The payload contains no spaces, so `new URL()` leaves it unencoded, and it remains a syntactically valid HTTPS URL [1]. Nothing on the path compared the `authorization_endpoint` against the configured server's origin [1].

Pillar's report lists affected versions as 1.0.137 through 1.18.32 on Windows and WSL. macOS and Linux are not affected because their launchers do not pass the URL through a shell [1][4]. The two primary accounts describe the trigger slightly differently. Pillar describes exploitation as requiring no user interaction or consent beyond configuring the server [1]. The Hacker News, by contrast, reports that the browser step runs only when a developer starts the sign-in flow, for example with `opencode mcp auth`, and that configuring a server or opening a session does not reach it [4]. In either reading, the user takes no action that would look dangerous, since starting authentication with a server already in the configuration is a routine step. Pillar also notes that the server does not need to be malicious when it is configured: a later compromise, a hijacked registry entry, a typosquat, or an expired domain could each turn a trusted entry hostile [1].

How the flaw was handled affects defenders as much as the flaw itself. Pillar reported privately through a GitHub Security Advisory on August 29, 2026. Anomaly acknowledged the report on September 25 and said malicious MCP servers were outside its threat model. The same day, it merged pull request #51414, whose description characterized the change as a consolidation of browser-launch call sites [1][7]. Version 1.18.33, published on September 28, bumped `open` to 11.0.4, which wraps the target in single quotes so PowerShell does not expand it. The release also added an `openUrl()` helper and a check in `redirectToAuthorization` that rejects non-HTTP(S) schemes [1]. Pillar and The Hacker News both note that the scheme check would not have stopped the published payload, and that the dependency upgrade is the change that closed the hole [1][4].

Anomaly requested no CVE and published no advisory. Pillar proposed a CVSS v3.1 score of 9.8 (CWE-78), but no official scoring exists [1]. That score assumes no user interaction. If, as The Hacker News reports, the flaw is reached only when a user starts authentication, a User Interaction: Required rating (approximately 8.8) would be more appropriate [4]; the analogous CVE-2025-6514 was itself scored with user interaction required [6]. Pillar's report gives the publication date as October 19 in its header and October 4 in its timeline, and secondary coverage uses October 19 [1][2][4]. According to The Hacker News, Pillar reported no signs of in-the-wild exploitation [4].

Copilot CVE-2026-103572: The Agent Rewrites Its Own Launch Configuration

Johann Rehberger's research on CVE-2026-103572 takes a different route to the same result [3]. Copilot in Visual Studio has an edit-confirmation routine, `EditsFunctionHandler.GetEditConfirmationTypeAsync` in `Microsoft.VisualStudio.Copilot.Core.dll`, that is supposed to require user approval before the agent changes sensitive files such as `.mcp.json` and IDE settings files. The check compared filename strings. Because the Win32 layer silently drops a trailing period, the agent could write to `.mcp.json.`, pass the check, and have the write land on `.mcp.json` [3]. Rehberger also documented a path traversal from the trusted Extensions directory using `..\` sequences. In SSMS, the traversal could read `SSMS.isolation.ini`, which exposes an installation ID; with that ID, an attacker could reliably build the per-installation settings path and modify it [3].

Rehberger described two exploitation chains, both beginning with indirect prompt injection delivered through a file the user opens or through output from an MCP server [3]. In the first, the agent uses the trailing-dot bypass to add an attacker-defined stdio server to `.mcp.json`, adds it to `enabledMcpServers`, and removes it from `disabledMcpServers`. The IDE then launches the attacker's command as an MCP server. In the second, which affects Visual Studio only, the agent rewrites the ripgrep path in `settings.VisualStudio.json` so that any Copilot search runs an attacker-chosen executable. The proof of concept launched `calc.exe`. Rehberger also noted that the model used in the tested configuration, `MAI-Code-1-Flash`, was weak against indirect prompt injection [3].

Microsoft rated the vulnerability Important, with a CVSS score of 8.8, and shipped a fix in the October 13, 2026 Patch Tuesday release [3][8]. According to Rehberger's timeline, he reported it on August 4, Microsoft confirmed it on September 16, and Microsoft judged it out of scope for a bounty [3]. From his analysis of the patch, Rehberger believes Microsoft added a `GetFullPath` call to normalize the path before validation, which strips trailing dots and resolves relative segments. This is the researcher's reading of the patch and not a statement from Microsoft [3]. Microsoft's Security Update Guide lists fixed builds of 17.14.42 for Visual Studio 2022 17.14 and 18.10.4 for Visual Studio 2026 18.10, and administrators should confirm build numbers for their channel there [8].

Rehberger's research affects Visual Studio 2026, older releases including Visual Studio 2022, and SSMS, which share the vulnerable component [3]. Microsoft's affected-product list, however, names only the two Visual Studio versions, even though the CVE title references SSMS [8]. SSMS administrators should therefore verify update availability for their build directly with Microsoft rather than assume a listed SSMS update exists. Microsoft reported no in-the-wild exploitation at the time of writing [8].

The Common Pattern

The two flaws reach the launch path from opposite directions, but they share one property: the launch path trusted data it should not have trusted. In opencode, a remote server controlled the data, while in Copilot an agent under adversarial influence controlled it. The table below summarizes the differences. Note that the two severity figures rest on different assumptions about user interaction and should not be compared directly.

Dimension	opencode MCP OAuth injection	Copilot CVE-2026-103572
Affected products	opencode 1.0.137–1.18.32 on Windows/WSL [1]	Copilot in Visual Studio 2026, earlier VS releases, and SSMS [3]
Attacker position	Malicious or compromised remote MCP server [1]	Indirect prompt injection via opened file or MCP output [3]
Launch path abused	Browser-launch helper (<code>open</code> → PowerShell) [1]	<code>.mcp.json</code> stdio server definition; ripgrep path setting [3]
Control bypassed	None existed: no origin binding, unescaped URL [1]	Filename protection bypassed by trailing dot and traversal [3]
Severity	Proposed CVSS 9.8 (researcher; assumes UI:N), no CVE [1]	CVSS 8.8 (UI:R), Important [3][8]
Fix	1.18.33 (Sep 28, 2026): <code>open</code> 11.0.4, scheme check [1]	October 13, 2026 Patch Tuesday; path normalization per researcher [3]
Vendor disclosure	No advisory or CVE; fix described in PR as a refactor [1][4]	CVE issued; no bounty awarded [3]

Taken together, the two cases suggest that MCP clients often treat configuration and discovery metadata as inert settings, even though they function as command lines. A stdio server entry is a command the client will run, and an OAuth authorization endpoint becomes a command whenever a helper passes it through a shell. Controls placed around these values, such as approval prompts, string allowlists, and scheme checks, fail when they validate a representation different from the one that is finally executed. In opencode, the scheme was checked but the shell interpreted the string. In Copilot, the filename was

checked but the filesystem normalized it. This resembles the "trust handoff" pattern CSA described in July 2026 for coding-agent sandbox escapes, in which an agent-written artifact is later consumed by an unsandboxed component that does not re-validate it [10].

Windows appears in both cases. This may reflect platform behaviors, such as PowerShell subexpression evaluation and Win32 path normalization, that cross-platform developers test less thoroughly. Pillar explicitly recommends testing the protocol surface on Windows, and Rehberger urges closer attention to Windows canonicalization issues [1][3]. Neither flaw required a sophisticated payload: once the launch path accepted the data, a short subexpression or a single trailing period was enough.

The disclosure handling also matters for risk management. A vendor that classifies malicious MCP servers as out of scope is stating an assumption that the user vetted every server and that the server stays trustworthy for as long as it is configured. Anomaly's published security policy states that external MCP servers a user configures are outside its trust boundary [4][9]. That position is coherent, but it moves the full burden of server vetting onto the user, and the absence of a CVE leaves security teams without the usual signal that a patch is needed. Organizations should account for that gap when they approve agentic developer tools.

Recommendations

Immediate Actions

Organizations should first upgrade opencode to 1.18.33 or later on every Windows and WSL host and apply the October 2026 security updates to Visual Studio [1][3]. For SSMS, which does not appear in Microsoft's affected-product list, teams should confirm with Microsoft which update covers their build [8]. Because the opencode fix has no CVE, endpoint and software inventory tools need a direct query for the `opencode-ai` package version, either through a global npm listing or the tool's own version output, rather than reliance on vulnerability feeds [1][2]. Where an immediate opencode upgrade is not possible, teams should remove unvetted remote MCP servers from the configuration; running `opencode mcp list` shows which servers are present [4]. For Visual Studio, teams should confirm installed build numbers against Microsoft's Security Update Guide [8].

Security operations teams should also check for signs of prior abuse. Useful starting points include unexpected entries in `.mcp.json` and IDE MCP settings, changes to `enabledMcpServers` or executable-path settings such as the ripgrep path in `settings.VisualStudio.json`, and process trees in which `devenv.exe`, `ssms.exe`, or the opencode process spawns `powershell.exe` with `-EncodedCommand` or other unexpected children. Depending on

installation method, the opencode parent may appear as `opencode.exe`, a Bun process, or a Node.js launcher, so detection rules should not key on a single image name. These are suggested hunting heuristics and not published indicators of compromise. Neither Pillar [4] nor Rehberger [3] published indicators of compromise.

Short-Term Mitigations

Over the next several weeks, organizations should manage MCP server configuration as a controlled artifact. Approved server lists should be maintained centrally, remote servers should be pinned to known origins, and changes to MCP configuration files should be monitored with the same scrutiny applied to CI pipeline definitions. Agent-authored changes to `.mcp.json`, IDE settings, task definitions, and similar files should require human review before they take effect. The CSA trust-handoff note makes the same recommendation for agent-written hooks, Git configuration, and task files [10].

Teams that build or extend MCP clients should adopt the engineering practices both researchers describe. URLs and other external strings should be passed to processes as argument vectors and never interpolated into shell command strings. OAuth discovery results should be bound to the configured server origin, and every server-supplied field should be treated as untrusted input [1]. File-access decisions should be made on canonicalized paths that match what will actually be written, with tests that cover Windows-specific behaviors such as trailing dots and extended-length `\\?\` prefixes [3]. CSA recommends extending those tests to trailing spaces and 8.3 short names, which raise the same canonicalization concerns. Projects that depend on the `open` package should upgrade to 11.x, although Pillar notes that this upgrade is a mitigation, not a reason to keep passing remote strings to it [1].

Strategic Considerations

The larger lesson is about where controls live. Rehberger argues that security controls for coding agents must hold even when the agent itself is compromised or misaligned [3]. Approval prompts implemented in the same code path the agent drives are only as strong as that code's input handling. Organizations should therefore put enforcement outside the agent where possible, through OS-level file-integrity controls on launch configuration, sandboxing that keeps MCP subprocesses from inheriting full developer credentials, and egress controls that limit which remote MCP origins a client can reach.

Procurement and vendor-risk processes should also ask directly how each AI developer tool vendor scopes MCP server trust and whether it issues CVEs for flaws that require a malicious server. A vendor that excludes malicious servers from its threat model is not necessarily negligent. Its customers, however, need compensating controls and their own monitoring, because the vendor's disclosure channel may not carry the signal they need.

CSA Resource Alignment

The opencode and Copilot findings extend a line of CSA AI Safety Initiative research on coding agents as an attack surface. CSA's July 2026 research note *AI Coding Agent Sandbox Escapes: The Trust Handoff Flaw* argued that coding-agent sandboxes generally held, while attacks succeeded where an unsandboxed component consumed agent-written output without re-validation [10]. CVE-2026-103572 fits that pattern closely. The agent writes `.mcp.json`, and the IDE's server launcher executes it without re-validation, although here the bypassed boundary is an in-process approval check rather than a sandbox. The note's guidance to inventory every component that consumes agent output and to review agent-authored configuration like source code applies directly.

CSA's June 2026 report *AI Coding Agents as Attack Surface: MCP, Poisoning, and Miasma* documented MCP auto-execution flaws and repository configuration poisoning, in which files such as `.vscode/settings.json` and agent settings files take effect when a repository is opened [11]. The Copilot chain shows that poisoned configuration does not need to arrive in a cloned repository, because a prompt-injected agent can write it at runtime. The opencode flaw adds a remote, server-originated variant that report's taxonomy should account for. CSA's research note *Agentjacking: MCP Injection via AI Coding Agents* covers the related case in which data returned from an MCP server hijacks a coding agent into running attacker code [13]. That is one of the delivery vectors Rehberger identifies for the Copilot chain [3].

CSA's May 2026 note *Sandboxing Agentic AI: Least-Privilege Patterns for MCP and Coding Agents* observed that MCP servers commonly run as stdio subprocesses with the calling user's credentials, and it recommended capability scoping and isolation for both MCP servers and coding agents [12]. Both flaws in this note gave the attacker that same ambient user context, which supports the earlier note's case for running MCP launches inside isolated, least-privilege environments rather than directly on developer workstations.

At the control-framework level, the CSA AI Controls Matrix (AICM) v1.1 provides the governance mapping [14]. Its Application and Interface Security (AIS) domain covers input validation and secure engineering of agent integrations. Its Threat and Vulnerability Management (TVM) domain covers the inventory and patch-verification practices needed when a vendor issues no CVE. Its Supply Chain Management, Transparency, and Accountability (STA) domain supports formal vetting and pinning of third-party MCP servers.

References

- [1] Eilon Cohen, Pillar Security. "[Open and Shut: Remote Code Execution in opencode via MCP OAuth.](#)" Pillar Security, October 19, 2026.
- [2] Kevin Poireault. "[Researchers Warn OpenCode Quietly Patched Critical RCE Flaw.](#)" Infosecurity Magazine, October 20, 2026.
- [3] Johann Rehberger (wunderwuzzi). "[Visual Studio Copilot and SSMS Remote Code Execution \(CVE-2026-103572\).](#)" Embrace The Red, October 13, 2026.
- [4] Swati Khandelwal. "[OpenCode Flaw Lets Malicious MCP Servers Execute Code During Sign-In on Windows and WSL.](#)" The Hacker News, October 22, 2026.
- [5] JFrog Security Research. "[mcp-remote Command Injection RCE \(JFSA-2025-001290844\).](#)" JFrog, July 9, 2025.
- [6] NIST. "[CVE-2025-6514 Detail.](#)" National Vulnerability Database, July 9, 2025.
- [7] Anomaly. "[opencode Pull Request #51414.](#)" GitHub, September 25, 2026.
- [8] Microsoft Security Response Center. "[Security Update Guide: CVE-2026-103572.](#)" Microsoft, October 13, 2026 (revised October 16, 2026).
- [9] Anomaly. "[opencode Security Policy.](#)" GitHub, accessed October 2026.
- [10] Cloud Security Alliance AI Safety Initiative. "[AI Coding Agent Sandbox Escapes: The Trust Handoff Flaw.](#)" Cloud Security Alliance, July 22, 2026.
- [11] Cloud Security Alliance AI Safety Initiative. "[AI Coding Agents as Attack Surface: MCP, Poisoning, and Miasma.](#)" Cloud Security Alliance, June 28, 2026.
- [12] Cloud Security Alliance AI Safety Initiative. "[Sandboxing Agentic AI: Least-Privilege Patterns for MCP and Coding Agents.](#)" Cloud Security Alliance, May 30, 2026.
- [13] Cloud Security Alliance AI Safety Initiative. "[Agentjacking: MCP Injection via AI Coding Agents.](#)" Cloud Security Alliance, June 2026.
- [14] Cloud Security Alliance. "[AI Controls Matrix \(AICM\) v1.1.](#)" Cloud Security Alliance, 2026.