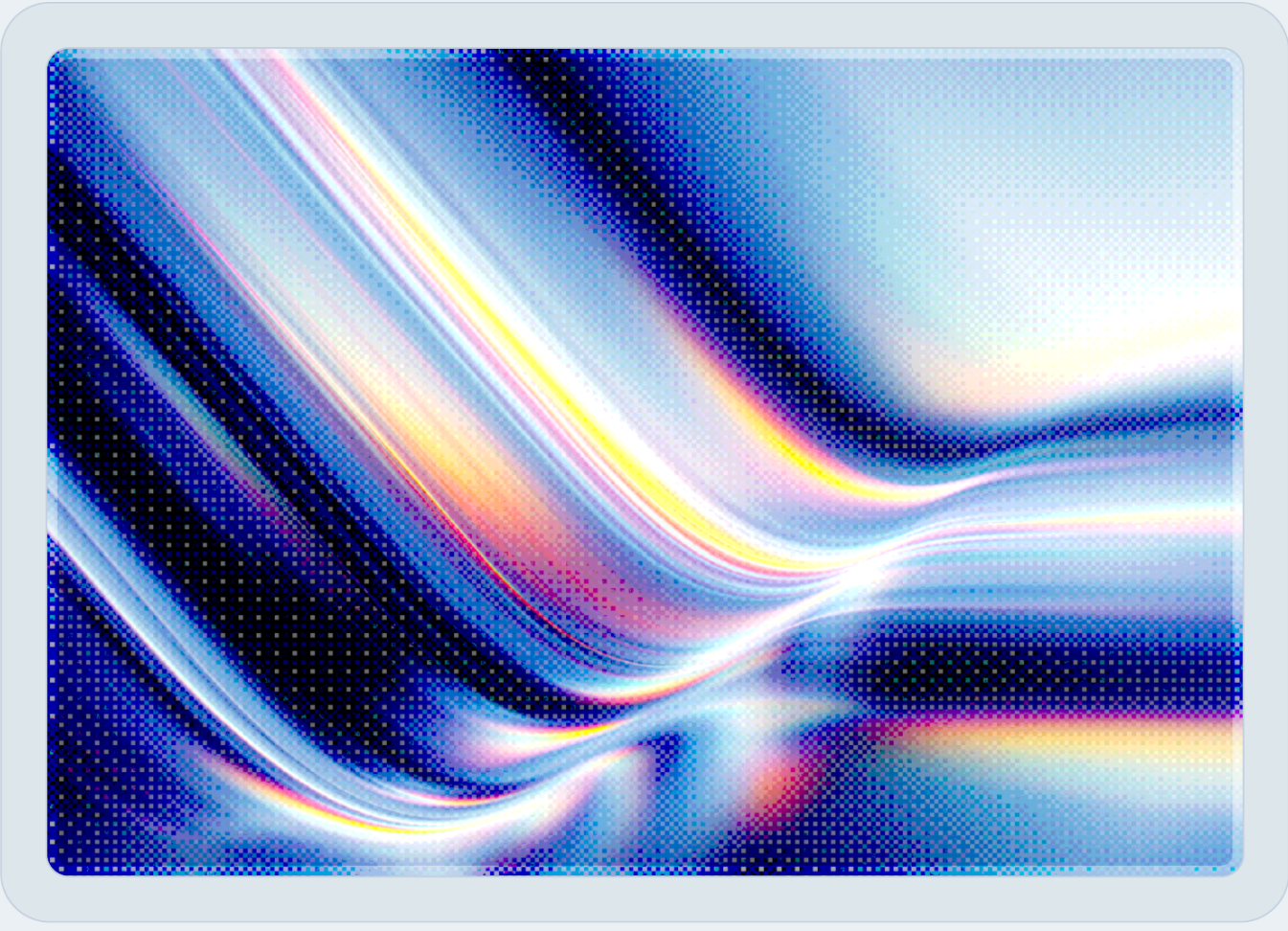
CSAI Foundation | Cloud Security Alliance

ThisBash: Browser Sessions as Remote Shells via Claude Dispatch

Security Implications and Guidance for Cross-Device AI Agent Sessions

2026-11-07

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

ThisBash is the name security researcher Gal Weizman of Forever Security gave to a design weakness in Anthropic's Claude Dispatch feature. According to his research, which The Register reported on November 5, 2026, the Dispatch interface and API could be reached from an ordinary logged-in claude.ai web session. As a result, any browser extension able to script that page could send instructions to the Claude Desktop agent paired with the account [1]. In the proof of concept, a script auto-accepted Claude's approval prompts and drove the desktop agent to install a malicious npm package, achieving code execution on the host [1][2]. Anthropic reportedly accepted the report as valid in May 2026 but closed it as a duplicate of an issue it was already tracking. It issued no CVE and said it planned to retire Dispatch by the end of October, yet the feature remained available to existing users as of late October [1] [3].

The most important lesson is architectural. In this design, the browser tab became a control plane for the desktop: when web, mobile, and desktop clients share one agent session, the security of local command execution can fall to the security of the least-trusted client, which is typically a browser with third-party extensions. Because Dispatch is designed to drive a desktop from another device, the exposure can also cross device and organizational boundaries, so that a compromised personal browser reaches a corporate laptop signed in to the same account [1]. The attack also clicked "Allow" automatically, which shows that in-page approval prompts are not a reliable control against scripted adversaries. Approvals rendered outside the browser and bound to a specific action are a different matter. CSA documented the same approval-forgery pattern in its July 2026 analysis of Claude for Chrome [4].

Retiring Dispatch may not remove the risk. The researcher states that the same session-sharing design persists in Claude Code Remote Control and Claude Cowork, although these claims have not been independently reproduced [1][2]. Anthropic's documentation also indicates that the feature is available on Team and Enterprise plans where the "Chat and Cowork unified" setting is enabled, so enterprise tenants should not assume they are out of scope [5]. Finally, the reporting describes no in-the-wild exploitation as of this writing, and no vendor fix specific to this issue has been documented [2].

Background

Anthropic introduced Dispatch in March 2026 as a research preview within Claude Cowork, its agentic desktop work mode. Dispatch lets a user assign tasks from a phone to a Claude agent running on their computer [6][5]. Anthropic's support documentation describes the capability plainly: the phone "can trigger real actions" on the desktop, including reading, moving, or deleting local files, using connected services, controlling the browser, and running desktop applications. The documentation also warns that manipulated instructions or malicious content can produce consequences that are "difficult or impossible to undo" [5]. Connectors, plugins, and file access configured in Cowork carry over to the remote surface without separate setup. Dispatch is therefore not a messaging feature that happens to reach an agent. It is a remote execution channel into a privileged local process, and it inherits whatever authority that process holds.

Weizman's insight was that the remote end of this channel was not limited to the mobile app. The Dispatch page and its supporting API were reachable from the standard claude.ai web application using the same authenticated session [1]. That shifted the question from "who holds the user's phone" to "what code can run inside the user's claude.ai tab." Modern browsers allow extensions with the appropriate host permissions to register content scripts that run in a page's main JavaScript world, where they can call the page's own APIs with the user's cookies [7]. Such scripts can also dispatch synthetic click and keyboard events, which a page will accept unless it checks whether the event was user-generated [4][8]. The population of code meeting that bar includes any extension the user installed and granted access to claude.ai or to all sites. It also includes any legitimate extension whose update channel or developer account an attacker compromises.

As of October 30, 2026, Anthropic's support page listed Dispatch as a limited beta on some Pro, Max, and Ultra plans. The page said the feature was unavailable to new users while existing users could continue to use it [1][5]. On Team and Enterprise plans, the support page says Dispatch is available where the "Chat and Cowork unified" setting is enabled, with Team plans in beta [5]. In parallel, Anthropic has extended Cowork to the web and mobile apps with sessions shared across clients. The researcher argues this arrangement reproduces the same trust relationship even without Dispatch being enabled [1]. Public reporting does not specify the operating system used in the proof of concept. Claude Desktop runs on macOS, Windows x64, and Linux [5], and CSA has not confirmed whether the behavior differs across platforms.

Security Analysis

The Reported Attack Chain

The publicly described attack proceeds in a short sequence of steps, most of which use functionality working as designed. That is what makes the issue primarily a design problem rather than a conventional implementation bug. The following table summarizes the chain as reported by The Register and secondary coverage [1][2]. CSA has not independently reproduced it.

Stage	Reported Action	Trust Assumption Exploited
1. Foothold	Extension registers a script that runs in claude.ai's main JavaScript context with the user's session	Any code in the claude.ai origin is the user
2. Concealment	Script opens the Dispatch page in a tiny, unfocused popup window	User will notice unexpected agent activity
3. Targeting	Script queries the session list to obtain the ID of the victim's Claude Desktop agent	Session enumeration is harmless to a legitimate client
4. Tasking	Script submits a series of prompts, rephrasing persistently until the agent complies	Model refusals are a security boundary
5. Approval	Script clicks "Allow" on approval prompts roughly once per second	An approval click reflects human intent
6. Execution	Agent runs a global npm install whose preinstall hook executes the payload	Package installation is a routine developer action

The demonstrated payload opened the calculator app, a conventional proof of execution. The Register notes that a real attacker would more plausibly target source code or cloud credentials, or install a persistent backdoor [1]. Two elements of the chain deserve particular attention. First, the attack does not depend on a prompt injection in the classic sense, where untrusted data is mistaken for instructions. The attacker speaks to the agent directly through the user's own authenticated channel, so at the authentication layer the agent receives no signal that distinguishes the attacker from the account holder. Second, the model's own reluctance served as friction rather than a barrier. Coverage of the research

describes an agent that refused at first and was then persuaded through repetition and rephrasing [1][9]. In this demonstration, model alignment affected how many attempts the attacker needed rather than whether the attack succeeded.

Why the Approval Layer Did Not Hold

Human-in-the-loop approval is the main runtime safeguard described in Anthropic's Dispatch documentation: Claude sends a notification when it needs the user's go-ahead [5]. ThisBash shows that this safeguard rests on an unstated assumption that the approval surface is reachable only by the human. When the approval button lives in a web page that hostile script can also reach, a click is just a DOM event. The script can generate that event as easily as the user can. CSA's July 2026 analysis of a separate Claude for Chrome flaw reached the same conclusion. In that case, Manifold Security showed that a co-installed extension could forge the approval click that launches an agent task, because the handler did not check whether the click came from a trusted user gesture [4][8]. In both cases, the approval flow functioned as a prompt for user deliberation but did not authenticate that the response came from the user.

The broader implication is that approval prompts in agentic products often serve as user-experience controls, and organizations should not count them as security controls unless they meet specific conditions. The approval must be rendered in a context that untrusted code cannot script, such as a native desktop dialog, an OS-level prompt, or a separate authenticated device. It must also be bound to the specific action being approved. Approval prompts that can be satisfied programmatically offer little assurance, and where agent tasks generate frequent prompts, approval fatigue further weakens their value for legitimate users.

Cross-Device and Cross-Organization Blast Radius

On a single machine, ThisBash is a sandbox escape: code confined to a browser extension's privileges gains the ability to run arbitrary processes through the desktop agent [1]. Dispatch's purpose, however, is to cross devices, which is what extends the blast radius. The browser running the malicious extension and the desktop running Claude need not be the same computer. They need only share a Claude account. An employee who uses a personal Claude account on a home computer full of unvetted extensions, and who also signs in with that account on a work laptop, creates a path from an unmanaged endpoint to a managed one [1]. That command path never passes through corporate extension allowlists or endpoint browser policy, and it arrives over the same channel as legitimate Claude traffic.

Forever Security's telemetry suggests that one part of the precondition, personal Claude accounts on corporate devices, is widespread in the organizations it observed. Across six organizations it examined, personal accounts made up between 18% and 71% of the Claude accounts observed in corporate environments, as reported. The worked examples given are 255 of 1,233 accounts in the low case, which computes to about 21% rather than the reported 18%, and 20 of 28 in the highest case [1]. Weizman described the six organizations as a small subset of Forever Security's customers [1], and secondary coverage notes that a security vendor's customer sample does not establish how common the pattern is across enterprises [2]. Readers should also weigh that the same firm supplied both the vulnerability research and the telemetry. The figures still indicate that many enterprises cannot currently assume that the Claude identity on a corporate device is a corporate-governed identity. Where that assumption fails, conventional extension management on corporate browsers offers no protection, because the attacker's foothold sits on a device the organization does not manage.

Status of Remediation and Residual Risk

Anthropic's handling of the report leaves security teams without a clear remediation signal. The company reportedly closed the submission as a duplicate, issued no CVE or bounty, and pointed to Dispatch's planned retirement rather than a fix. It did not respond to The Register's questions about why the retirement had not occurred on schedule [1]. The Register's headline characterizes Anthropic as declining to fix the flaw, while secondary coverage frames Anthropic's response as retiring the feature, and WindowsForum found no Anthropic statement responding to the specific allegation [2][9]. Both framings are consistent with the available facts: Anthropic has acknowledged the issue but has not published a mitigation.

More consequential for planning is the researcher's claim that the design pattern outlives Dispatch. Cowork is now accessible on the web and shares sessions with the desktop app, and Claude Code Remote Control similarly allows a remote client to drive a local coding agent. Weizman asserts that an extension could direct those sessions to access files or install packages in the same way [1]. Public reporting does not include test results for those features, and CSA treats these claims as plausible but unverified. The architectural pattern is general, however. Any product that allows a web client to issue commands to a locally privileged agent session, protected only by the web session cookie and an in-page approval, presents the same class of exposure. The researcher contrasts this with OpenAI's Codex, which he says does not expose desktop-initiated sessions to the web client; CSA has not verified this comparison. The proposed remediation is to treat the web client as a separate, lower-trust device that cannot authorize sensitive actions such as command execution [1].

Recommendations

Immediate Actions

Organizations should first determine whether the precondition exists in their environment. Security teams should identify Claude Desktop installations through endpoint software inventory, and determine Dispatch or Cowork use through user survey or Anthropic admin console data where available. Where Dispatch or remote Cowork control is not a business requirement, it should be disabled. WindowsForum, citing a Harmonic Security practitioner guide, reports that no centralized administrative toggle specific to this issue has been documented, so for individual plans this step may depend on user action and policy communication [2]. On Team and Enterprise plans, administrators should review whether "Chat and Cowork unified" is enabled for their tenant and disable it where cross-device execution is not required [5]. In parallel, teams should audit managed browsers for extensions that hold host permissions for claude.ai or for all URLs, and remove those that are not explicitly approved. That covers the corporate side of the exposure; the personal-account side requires a policy response.

Organizations should also instruct staff not to sign in to Claude Desktop on corporate devices with personal Claude accounts. Where an enterprise Claude plan exists, its use should be required on managed endpoints, which brings identities under organizational governance. This step alone does not close the exposure, since an enterprise account with "Chat and Cowork unified" enabled may present the same session-sharing risk. Where no enterprise plan exists, teams should weigh restricting Claude Desktop on high-value endpoints, such as developer workstations with cloud credentials, until the session model changes. If unexplained agent activity has occurred, such as unexpected global npm installs or Claude sessions the user does not recognize, teams should treat the endpoint as potentially compromised and rotate credentials accessible from it.

Short-Term Mitigations

Over the next several weeks, organizations should add detection for the observable artifacts of this attack class. Useful signals include package installation and script execution spawned by the Claude Desktop process tree, particularly package-manager lifecycle hooks. Outbound connections from processes the desktop agent launched are another signal. On managed browsers, organizations should enforce extension allowlisting through browser enterprise policy and restrict host permissions for AI application domains. Policies should also cover extension updates, since a trusted extension acquired or compromised by an attacker is as dangerous as a malicious one.

Organizations should reassess any internal risk acceptance that relies on agent approval prompts as a compensating control. For high-risk actions, including command execution, package installation, credential access, and outbound data transfer, approval should occur in a channel that untrusted browser code cannot reach. The desktop agent's file and connector configuration should also be scoped to least privilege, so that a successful hijack yields less. Finally, procurement and vendor-risk teams should ask Anthropic, and any other vendor offering cross-device agent sessions, for written answers to three questions: which clients can initiate actions on a local agent, whether administrators can restrict that capability by client type, and whether approval events are authenticated against the originating device.

Strategic Considerations

ThisBash is one instance of a pattern CSA expects to recur as agent products converge on "start anywhere, run anywhere" session models. The underlying lesson is that agent sessions need device-aware authorization, not only user-aware authentication. A session cookie proves which account is speaking. It does not prove which device is speaking, whether that device is managed, or whether a human or a script produced the request. Enterprise architectures should treat each client type as a distinct principal with its own privilege ceiling. Under that model, web clients would be unable to authorize local execution by default, and approvals would be cryptographically bound to a trusted device.

This also argues for applying Zero Trust principles inside the agent product itself, not just around it. The browser should be modeled as a hostile environment that shares an origin with untrusted extension code. Vendors should publish their session trust models and offer administrators controls that disable cross-device execution for enterprise tenants. Organizations should include agent session-sharing in their AI threat models alongside prompt injection and tool poisoning. Finally, the handling of this report, closed as a duplicate without a CVE or a published advisory, reflects a gap CSA has observed in how AI agent design flaws are disclosed. In the earlier Claude for Chrome case, the tracking issues were reportedly marked resolved without a functional code change [4]. Without an identifier or advisory, defenders have little to track, and enterprises have limited contractual means to obtain timely answers.

CSA Resource Alignment

The most directly related CSA work is the July 2026 research note *Claude for Chrome: Synthetic Click Flaw Lets Extensions Hijack AI Actions* [4]. That note analyzed how a co-installed extension could forge the approval click that authorizes a Claude agent action in the browser, and it recommended auditing extensions with access to claude.ai and disabling unattended modes. ThisBash extends that finding in two

directions: the forged approvals now authorize native command execution rather than browser actions, and the target can be a different physical machine. The recommendations in the earlier note, particularly extension auditing and skepticism toward single-click approvals, apply directly and with greater urgency.

CSA's June 2026 research note *AutoJack and Agentjacking: AI Agent Frameworks as a New RCE Attack Surface* [10] documents a related trend in which AI agent frameworks become remote code execution vectors. In AutoJack, a malicious web page reached a localhost agent service; in Agentjacking, poisoned telemetry data hijacked coding agents. ThisBash fits the same pattern of web-originated input reaching a privileged local agent. It differs in that the attacker uses the user's authenticated session rather than smuggling instructions through content. Where AutoJack exploited weak origin checks on a local service, ThisBash passes every origin check, because the hostile code already runs inside the trusted claude.ai origin.

For control mapping, the CSA AI Controls Matrix (AICM) v1.1 [11] provides the governing control set. Its Identity and Access Management (IAM) domain addresses the need to authenticate and authorize distinct client principals. Its Threat and Vulnerability Management (TVM) domain covers how agent design flaws should be tracked even without a vendor CVE, and its Supply Chain Management, Transparency, and Accountability (STA) domain applies to the browser extensions and npm packages that both enabled and delivered this attack. CSA's MAESTRO agentic threat modeling framework [12] offers a structure for analyzing the issue across layers. Here, the weakness spans the agent ecosystem layer, where multiple clients share a session, and the deployment and infrastructure layer, where the desktop agent holds host privileges. Organizations threat-modeling cross-device agent products should add client-type spoofing and approval forgery as explicit threats within that framework.

References

[1] The Register. "Anthropic won't fix a Claude Dispatch flaw that could let attackers execute remote commands across corporate boundaries." The Register, November 5, 2026.

[2] WindowsForum. "ThisBash: Reported Claude Dispatch Flaw Could Let Browser Extensions Run Code on Desktops." WindowsForum, November 5, 2026.

[3] Mezha. "Claude Flaw Lets Browser Extensions Execute Code on PCs, Anthropic Has Left It Unfixed for Six Months." Mezha, November 6, 2026.

[4] Cloud Security Alliance. "Claude for Chrome: Synthetic Click Flaw Lets Extensions Hijack AI Actions." CSA AI Safety Initiative, July 18, 2026.

[5] Anthropic. "Assign tasks from anywhere in Claude Cowork." Claude Help Center, accessed November 7, 2026.

[6] Storyboard18. "Anthropic introduces 'Dispatch' feature, turning Claude into a remote AI assistant." Storyboard18, March 19, 2026.

[7] Google. "chrome.scripting API Reference." Chrome for Developers, accessed November 7, 2026.

[8] BleepingComputer. "Claude Chrome extension flaw lets malicious extensions trigger AI actions." BleepingComputer, July 16, 2026.

[9] DigitBin. "Claude Dispatch Flaw Lets Extensions Run Commands on Your PC." DigitBin, November 6, 2026.

[10] Cloud Security Alliance. "AutoJack and Agentjacking: AI Agent Frameworks as a New RCE Attack Surface." CSA AI Safety Initiative, June 23, 2026.

[11] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, June 22, 2026.

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