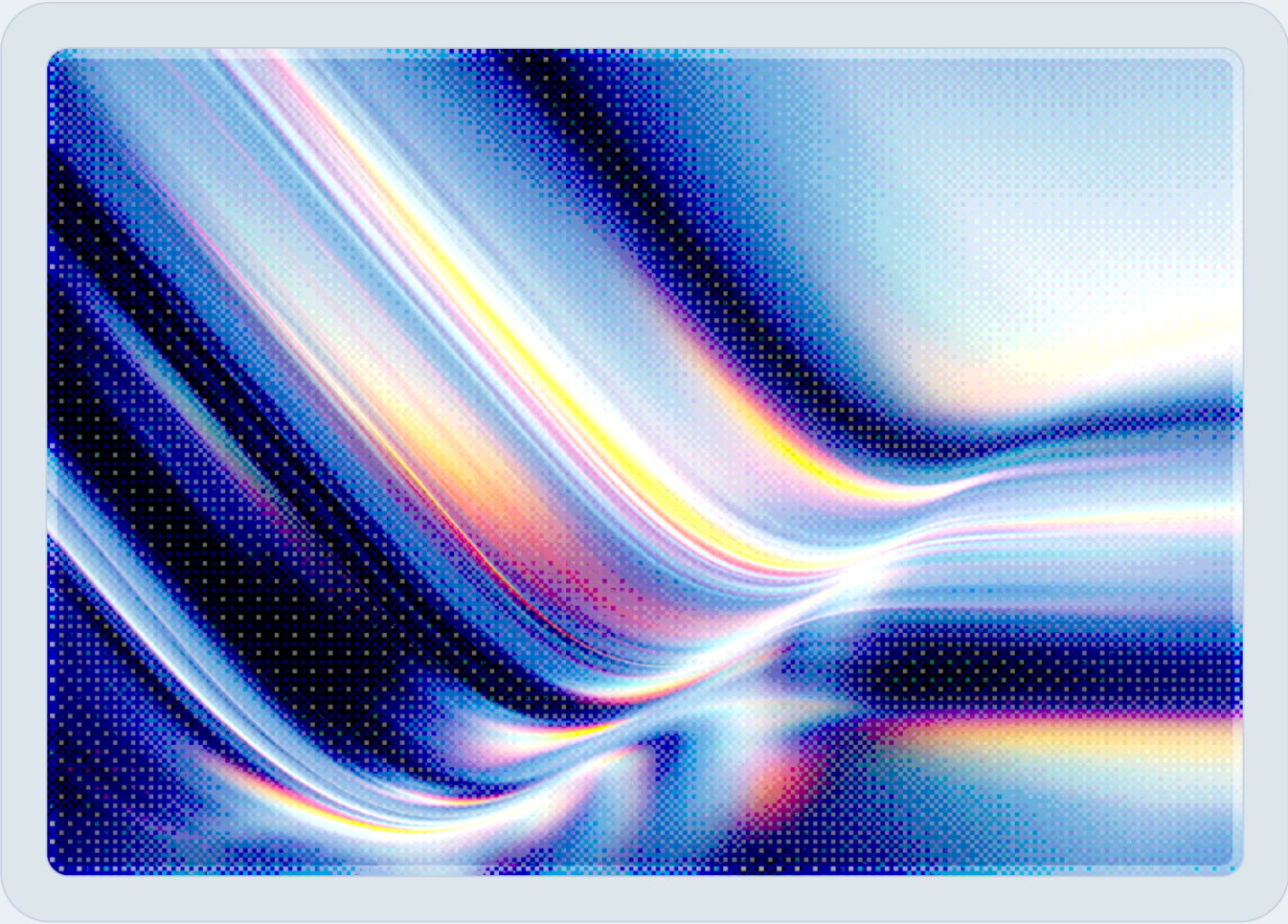
CSAI Foundation | Cloud Security Alliance

BragJack: One Extension Hijacks Five Browsers' AI Agents

2026-09-22

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Security researcher Gal Weizman of Forever Security disclosed BragJack on September 16, 2026, demonstrating that a single, ordinary-looking browser extension could hijack the built-in AI agents of five separate agentic browsers: Google Chrome's Gemini Live, Perplexity Comet, Microsoft Edge Copilot Actions, Opera Neon, and Anthropic's Claude in Chrome [1][2].
- Unlike prompt injection, which tricks a model with cleverly worded text, BragJack uses a technique Weizman calls "Prompt Forcing": the extension abuses ordinary, already-granted extension permissions to hand the agent an attacker-authored instruction directly, without needing to bypass any model-level guardrail [1][3].
- CSA's analysis: the pattern across all five cases points to an architectural gap rather than a single coding mistake — each browser vendor built a privileged communication channel between trusted first-party web pages and its AI agent, then failed to fully wall that channel off from the separate, well-documented permissions Chromium already grants to any installed extension, such as `declarativeNetRequest` (DNR) header and redirect rewriting [1][2].
- Google and Microsoft assigned CVE identifiers to the flaws in their products — CVE-2026-0628 for Chrome's Gemini Live (CVSS 8.8) and CVE-2026-55945 for a related race condition in Microsoft Edge — and both companies have shipped fixes, in Chrome 143.0.7499.192 and Edge 150.0.4078.48 respectively [1][2][4].
- Depending on the browser, a successful BragJack chain could force the agent to read files and folders, exfiltrate browsing history, activate the camera or microphone, capture screenshots, read email, or take authenticated actions on logged-in sites, all without a conventional prompt injection payload or any visible confirmation dialog [2][3].
- Weizman collected roughly $20,000 in combined bug bounties across the five affected vendors, ranging from $7,000 (Chrome, Perplexity Comet) down to $600 (Anthropic), and all five have acknowledged and remediated the specific proof-of-concept behaviors demonstrated [1][2].

Background

Over the past year, several major browser vendors — including Google, Microsoft, Perplexity, Opera, and Anthropic — have shipped AI agents directly into the browser chrome: an assistant that can read the current page, reason about it, and — increasingly — act on the user's behalf inside authenticated sessions. These agents are typically implemented as a privileged internal component with elevated access to browser APIs, distinct from the ordinary web page sandbox that constrains regular JavaScript. To make that agent usable, vendors also had to build a communication path so that the agent's own first-party pages, chat interfaces, or side panels could issue it instructions. CSA's own analysis of a related flaw in Claude for Chrome earlier this year described this pattern as creating "a channel that the vendor never expected anyone but its own trusted code to use" [5].

BragJack demonstrates that this pattern recurs across the industry, not just in one product. Weizman's research, published as a technical write-up on Forever Security's blog and covered independently by BleepingComputer and other outlets, targeted five of the best-known agentic browser features and found each one exploitable using extension permissions Weizman characterizes as ordinary — meaning permissions that do not require an undisclosed zero-day capability, though not necessarily permissions every user grants without scrutiny [1][2]. The unifying mechanism, which Weizman's write-up terms "DiNneR Serving," pairs two capabilities that Chromium extensions have long been permitted to request: the ability to weaken a page's security headers (stripping Content-Security-Policy or X-Frame-Options protections via declarativeNetRequest) and the ability to redirect a script request to an attacker-controlled server. Combined, an extension can cause a browser's own trusted, first-party page to load attacker-controlled JavaScript inside a security context the agent implicitly trusts [1][2].

Because the underlying flaw sits in the trust boundary between the extension platform and the agent's privileged channel, BragJack did not require Weizman to defeat any model safety training or write a clever jailbreak prompt. Each vendor's specific implementation added its own twist. Chrome's Gemini Live channel had not applied its declarativeNetRequest restrictions to the WebView context the agent used, letting the redirect-and-strip technique reach a script the browser implicitly trusted [1][2]. Microsoft Edge exposed a private API intended for its own marketing pages and briefly toggled a privileged "enable AI tools" flag during a race condition that a synthetic, debugger-generated click could exploit before the agent re-checked its state [2]. Perplexity Comet had left a testing subdomain without the network protections applied to its production agent surface [2]. Opera Neon had not restricted content-script attachment to its own trusted communication domain at all [2]. Claude in Chrome's content script listened for clicks on task-triggering page elements without confirming the click was genuinely user-initiated — a variant of the same synthetic-click authentication gap CSA documented in a dedicated research note in July 2026 [2][5].

Security Analysis

The significance of BragJack is less about any single vendor's bug and more about what it reveals about a shared design assumption across the agentic browser category: that a browser's own first-party surfaces are inherently safe places for a privileged AI agent to receive instructions. That assumption breaks down the moment a user installs any third-party extension, because Chromium's extension platform appears to have been designed around an older threat model — one where extensions might misbehave against ordinary web content, but were not expected to reach a browser's own privileged internal channel. Weizman's report frames this directly: the attacker never needs to compromise the AI vendor's servers, the underlying language model, or the user's account credentials. A single, unremarkable extension already installed for an unrelated purpose is sufficient, which arguably lowers the bar compared to attacks that require social engineering a user into installing something obviously malicious [1][2].

The table below summarizes how the same underlying pattern manifested differently across the five affected products, illustrating that the vulnerability class, not the specific exploit steps, is what organizations should track.

Browser / Agent	Affected Mechanism	Demonstrated Impact	Vendor Status
Chrome — Gemini Live	DNR-based header stripping and script redirection inside agent WebView; CVE-2026-0628 (CVSS 8.8)	File access, screen capture, camera/microphone activation [1][2]	Fixed in Chrome 143.0.7499.192 [2]
Microsoft Edge — Copilot Actions	Race condition toggling an internal "enable agent tools" flag via synthetic click; CVE-2026-55945	Authenticated-site actions triggered without genuine user gesture [1][2]	Fixed in Edge 150.0.4078.48 [2][4]
Perplexity Comet	Unprotected testing subdomain allowing direct `chrome.runtime.sendMessage()` calls to the agent extension	Screenshots, browsing history, user data, agent-driven actions [1][2]	Remediated; bounty paid [1]
Opera Neon	No content-script restriction on Opera's own trusted communication domain	Email reading and message	Remediated; bounty paid [1]

Browser / Agent	Affected Mechanism	Demonstrated Impact	Vendor Status
		summarization exfiltration [2]	
Claude in Chrome	Content script accepted synthetic (non-`isTrusted`) clicks on task-triggering elements	Execution of predefined agent tasks without genuine user approval [2][5]	Remediated; bounty paid [1]

Two patterns are worth calling out beyond the individual fixes. First, the Edge and Claude in Chrome cases both involved synthetic clicks generated through the browser's own debugger or automation APIs, rather than any exotic exploitation technique. In CSA's assessment, this illustrates that "the user clicked to approve this" should not be treated as a meaningful security boundary unless click authenticity is verified at the code level. Second, the range of remediation quality varied: Google and Microsoft issued formal CVE advisories with version-pinned fixes [1][2][4], while Perplexity, Opera, and Anthropic acknowledged and remediated the specific proof-of-concept but did not publish comparably detailed advisories in the sources reviewed for this note [1][2]. Organizations evaluating agentic browser tools should not assume uniform maturity in vulnerability handling across vendors simply because all five ultimately paid a bounty.

Recommendations

Immediate Actions

- Inventory which browsers with built-in AI agents are installed across the organization's fleet — Chrome, Edge, Opera, or any browser bundling Comet or Claude in Chrome — and confirm each is running a version at or above the vendor's patched release (Chrome 143.0.7499.192; Edge 150.0.4078.48) [1][2].
- Audit installed browser extensions on any endpoint where an agentic browser feature is enabled, with particular attention to extensions requesting `declarativeNetRequest`, `webRequest`, or debugger-related permissions, since these are the specific capabilities BragJack's technique depends on.

- Disable "act without asking" or auto-approval settings for AI browser agents on high-sensitivity accounts until an organization has separately verified the specific browser version and extension inventory in use.

Short-Term Mitigations

- Extend extension allowlisting and permission-review policies, which most organizations already apply to traditional browser extensions, to explicitly cover the new risk that a permitted extension can reach a co-resident AI agent's privileged channel — do not treat AI agent security as separate from extension governance.
- Where feasible, restrict AI browser agent usage to dedicated, low-privilege browser profiles that do not carry authenticated sessions to sensitive services such as email, source code repositories, or financial accounts, bounding the blast radius of any future hijack.
- Require vendors of agentic browser tools to document, as part of security questionnaires, how their agent-to-extension trust boundary is enforced and whether synthetic or debugger-generated input events are distinguished from genuine user interaction.

Strategic Considerations

- Treat BragJack as confirmation that "agent hijacking through the extension surface" is now a recurring, cross-vendor category rather than an isolated implementation bug, following the same pattern CSA identified in its analysis of the Claude for Chrome synthetic-click flaw earlier in 2026 [5]. Enterprise AI governance programs should add this category explicitly to agentic browser risk assessments rather than waiting for the next branded disclosure.
- Push for browser and AI agent vendors to adopt cryptographic or otherwise unforgeable event-provenance checks (verifying `Event.isTrusted` or an equivalent signal) as a baseline requirement before granting an agent privileged action capability, rather than relying on UI-level approval dialogs that assume all clicks are genuine.
- Factor vendor responsiveness and fix quality — not just the existence of a bug bounty payout — into ongoing vendor risk scoring for agentic AI tools, since this disclosure showed measurable variance in how thoroughly different vendors addressed the underlying architectural issue versus the specific proof-of-concept.

CSA Resource Alignment

BragJack extends a pattern CSA's AI-assisted rapid research program has been tracking closely across 2026: browser-integrated AI agents concentrate privileged capability behind a communication channel that turns out to be reachable by adjacent, seemingly unrelated attack surfaces.

CSA's "Claude for Chrome: Synthetic Click Flaw Lets Extensions Hijack AI Actions" [5] is the most directly relevant prior artifact, since it examined the identical root cause — a co-resident extension forging a click to trigger privileged agent actions — in one of the same five products BragJack targets. That note's central finding, that Claude for Chrome's content script failed to verify `Event.isTrusted` before executing predefined tasks, is essentially the Claude-specific instance of the synthetic-click weakness BragJack also documents in Microsoft Edge's Copilot Actions, alongside three structurally different extension-to-agent trust gaps — header/redirect manipulation, an unprotected subdomain, and a missing content-script restriction — in the other three affected browsers. Its recommendation to extend extension allowlisting and permission-review processes to explicitly cover AI agents applies without modification to the wider set of affected products documented here.

CSA's "BioShocking: AI Browser Agents Weaponized for Credential Theft" [6] examined a structurally different but complementary attack path against overlapping products (including Claude in Chrome and Perplexity's Comet lineage): rather than exploiting an extension permission gap, BioShocking used indirect prompt injection to convince the agent's undifferentiated instruction/content stream that attacker-supplied text was a legitimate command. Read together, BragJack and BioShocking show that browser AI agents face at least two independent, non-overlapping routes to hijack — architectural extension-to-agent trust gaps and content-level prompt injection — and that closing one does not address the other.

In CSA's reading, both incidents map most directly to the AI Controls Matrix (AICM v1.1) [7], particularly its identity and access management and application/interface security domains, and to the MAESTRO agentic AI threat modeling framework's agent ecosystem layer [8], which explicitly models trust boundaries between an agent and the third-party components sharing its runtime environment. Organizations red-teaming agentic browser deployments should treat "can a co-installed extension reach the agent's privileged channel" as a standing test case, not a one-time check against a single disclosed CVE.

References

[1] BleepingComputer. "BragJack attacks hijack AI browser agents through malicious extensions." BleepingComputer, September 2026.

[2] Forever Security. "BragJack Technical Overview: How We Hijacked Top 5 Browsers' Internal Agents With Just One Single Extension." Forever Security, September 2026.

[3] The Hacker News. "One Extension Could Hijack AI Assistants Across Chrome, Comet, Edge, Opera Neon and Claude." The Hacker News, September 2026.

[4] WindowsForum. "CVE-2026-55945 Fixed in Edge 150.0.4078.48: AI Hijack Flaw." WindowsForum, September 2026.

[5] Cloud Security Alliance. "Claude for Chrome: Synthetic Click Flaw Lets Extensions Hijack AI Actions." Cloud Security Alliance, July 2026.

[6] Cloud Security Alliance. "BioShocking: AI Browser Agents Weaponized for Credential Theft." Cloud Security Alliance, June 2026.

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

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

