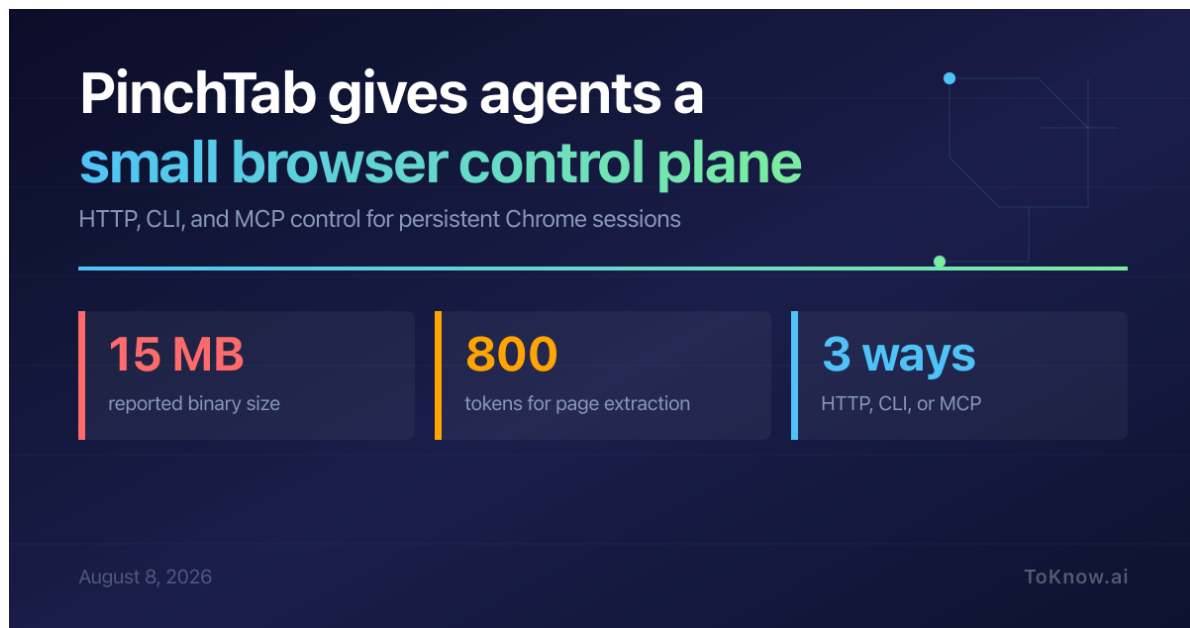


PinchTab: A Small HTTP Control Plane for Browser Agents

Kabui, Charles

2026-08-08

[Read at ToKnow.ai](#)



The graphic is a dark blue rectangular banner. At the top left, the text 'PinchTab gives agents a small browser control plane' is displayed in white and light blue. Below this, in smaller white text, is 'HTTP, CLI, and MCP control for persistent Chrome sessions'. To the right of the text is a faint, light blue geometric diagram consisting of a hexagon with internal lines and a small blue dot. Below the main text is a horizontal line with a small green dot. Underneath this line are three dark blue boxes with vertical colored bars on their left sides. The first box has a red bar and contains the text '15 MB' in red and 'reported binary size' in white. The second box has a yellow bar and contains the text '800' in yellow and 'tokens for page extraction' in white. The third box has a light blue bar and contains the text '3 ways' in light blue and 'HTTP, CLI, or MCP' in white. At the bottom left of the banner is the date 'August 8, 2026' in white, and at the bottom right is the text 'ToKnow.ai' in white.

PinchTab gives agents a small browser control plane
HTTP, CLI, and MCP control for persistent Chrome sessions

- 15 MB**
reported binary size
- 800**
tokens for page extraction
- 3 ways**
HTTP, CLI, or MCP

August 8, 2026 ToKnow.ai

[PinchTab](#) is a standalone Go server for agents that need to use Chrome without being tied to a browser-automation SDK. An agent can send HTTP requests, use the CLI, or connect through MCP, then inspect page text, accessibility snapshots, screenshots, and changes. Stable accessibility references give actions names that survive ordinary page movement better than screen coordinates. The server supports headed and headless profiles, persistent sessions,

multiple browser instances, and audits. Its README reports a roughly 15 MB self-contained binary and page extraction around 800 tokens, advertised as 5 to 13 times cheaper than screenshot-based interaction. Those figures are project-reported measurements, not an independent benchmark.

That small boundary is useful for teams building agents in different languages. A Python worker, a Rust service, and a desktop assistant can share a logged-in browser through the same HTTP interface, while persistent profiles avoid signing in for every task. The trade-off is authority: PinchTab can read pages and operate a real browser. It binds to loopback and limits browsing to local sites by default, while public browsing and remote access require deliberate security changes. Authentication, network exposure, stored sessions, and the optional CloakBrowser stealth path deserve review before this becomes shared infrastructure.

PinchTab points toward a simpler agent stack, but simplicity at the API boundary does not make browser control low-risk. The useful question is not whether an agent can click, but which identities, cookies, and networks that click can reach.

Sources:

- [PinchTab repository](#)
- [PinchTab security guide](#)
- [PinchTab core concepts](#)
- [PinchTab API documentation](#)

Disclaimer: For information only. Accuracy or completeness not guaranteed. Illegal use prohibited. Not professional advice or solicitation. **Read more:** [/terms-of-service](#)