Episode Book

EPISODE #6167

Streaming Browser Cookies to a Headless Server

PUBLISHED	20 October 2026 at 19:32 UTC
DURATION	20:17
LISTEN	myweirdprompts.com/episode/streaming-browser-cookies-server-01064/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.11
BOOK GENERATED	21 October 2026 at 07:50 UTC

SYNOPSIS

Daniel wants his agent-driven browser off his laptop and onto the home server, but the server has never logged into anything. Chrome closed the obvious path in March 2025 when version 136 started ignoring the remote debugging port on your default profile — infostealers had been using it to lift sessions. So what are the real options? This episode walks through the extension cookie API and CookieCloud, headless browsers like Steel and Browserless, the one-way sync problem, Device Bound Session Credentials (DBSC), and the unglamorous option that might win: just log in once by hand and use Chrome's Memory Saver.

TRANSCRIPT



Corn

Daniel wants his browser off his laptop and onto the home server, and the only thing in the way is that the server has never logged into anything. He drives his own Chrome with Claude through the extension, the agent inherits his cookies, and that means it can work inside services that have no API, no MCP server, and a command line tool that falls over every other Tuesday. It's the best part of his setup. The cost is memory, because agent tabs and windows pile up and the laptop starts begging. So, could the laptop's cookies be streamed across the house to a headless Chrome, live, and securely? Or is that dead on arrival?



Herman

Not dead. But the door everybody reaches for first has been bolted shut, and on purpose. In March of last year Chrome one thirty-six started ignoring the remote debugging port when it points at your default profile. That was the classic way to say to a running Chrome, hand me everything, cookies included. Infostealers had been using exactly that to lift sessions, so Google closed it.



Corn

So the obvious plan was already a crime scene before Daniel thought of it.



Herman

Pretty much. If you launch Chrome with a separate data directory, the port works again, but that profile uses a different encryption key and, more to the point, it's empty. It has none of his logins. Which is the whole problem.



Corn

To be fair to Chrome, I'd have shut it too. A debugging port on your real profile is a master key left on the hall table.



Herman

Google's advice for automation now is a separate build called Chrome for Testing, which has none of your life in it. That's the right design for a server, and it's useless for what Daniel wants, because what he wants is the life.



Corn

He wants the life without the laptop.



Herman

Here's what I'd like your read on, Corn. What is he actually getting from the cookies? Is it the logins themselves, or that the agent can walk into a service without him typing a code at it?



Corn

The second. For something with no API and no MCP server, a logged-in browser is the integration. There's nothing else to build. That's why he'd rather not lose it, and why a clean empty browser on the server does nothing for him.



Herman

That's my read too. It also means the bar is high, because anything that makes the server's browser half logged in is worse than nothing.



Corn

Fine, so the front door's locked. Is there a side door?



Herman

There is, and it's the extension API. A Chrome extension with the cookies permission can read cookies for any host it's been granted access to, and it gets an event every time one changes. Set, removed, expired. That's your stream. Not a file you copy, a feed.



Corn

Which is also what Claude in Chrome already is, sort of. An extension acting on his behalf.



Herman

In spirit, yes. This would be a smaller extension doing one job. Every time a cookie changes, it encrypts it and pushes it to something on the server. One event per change, and you replay each one into the server's browser.



Corn

And has anyone built that, or are we reasoning from the armchair?



Herman

Someone built it. CookieCloud. It's open source, GPL, you run it in a Docker container on your own box, and the extension in Chrome or Edge uploads your cookies and local storage to it, encrypted with AES. The key comes from an identifier and a password you choose, so the server mostly stores a blob. Another browser downloads and decrypts.



Corn

Mostly. That word's doing a lot.



Herman

It is. Two trade-offs. First, it's one-way. One browser uploads, the other downloads. That's fine for Daniel, the laptop is the source of truth, but if the server's browser logs in somewhere fresh, none of that flows back. Second, if you ask the server to hand you the decrypted cookies, you send it the password in the request, so the server sees it. I'd expect you could pull the encrypted blob and decrypt locally to avoid that, but I haven't checked that the tool supports it cleanly.



Corn

So the cookie vault is a container on a box that already holds half his keys.



Herman

Yes. That's the cost, said plainly. The moment the stream works, the server is a vault of working sessions for every site he includes. Anyone who gets into that box doesn't need his passwords, they just need the blob and the password that opens it. So you'd want it travelling over Tailscale or WireGuard rather than loose on the LAN, and encrypted at rest, and you'd keep the list of hosts short. The extension needs host permissions per site anyway, so you can't accidentally sync the entire jar.



Corn

Short list is the right instinct. I don't need the server knowing my recipe bookmarks are logged in.



Herman

Then the other half. Getting a cookie into a headless browser is the easy part. The automation protocols all let you set cookies into a browser context before you open a page. The question is what holds the browser. Steel Browser is Apache licensed, it runs in a container, it has an API and a debugger and even a little web interface. Its page says it keeps cookies and local storage across requests.



Corn

Across requests. Not across a restart.



Herman

That's what I can't tell you. The README doesn't say whether state survives a restart, and it doesn't describe an endpoint for injecting cookies, though it might exist. Browserless is the other one, and there the licence is the catch. It's SSPL unless you pay, and the persistent sessions feature, cookies and cache kept for weeks, is listed under the commercial tier. Whether the self-hosted build gives you any of that, I don't know.



Corn

So both of them are a box that runs Chrome, and you might be gluing the persistence yourself.



Herman

Likely, yes. And the glue is small. As far as I know, the standard automation protocol lets you set cookies on a browser that's already running, no restart, so a little service on the server receives each pushed event and applies it to the live context. A second or two behind the laptop, I'd guess. CookieCloud also carries local storage, which matters, because plenty of sites keep their tokens there and not in a cookie.



Corn

So you're not copying a profile, you're keeping two of them in step.



Herman

Which is where I get nervous, and I'll come to it.



Corn

We did a whole episode in August on cookies as credentials, one-off exports, how short the lifetimes are, no refresh token to speak of. This is the continuous version of that.



Herman

And the short lifetime is why a stream beats a one-time export. If a site rotates the session cookie every few hours, a snapshot goes stale, and a stream follows along. The nervous part is the other direction. Some sites hand you a fresh session value every so often, and invalidate the old one. If the agent on the server triggers that, the laptop's copy goes stale, and the next time the laptop pushes it may knock the server out. Two browsers, one login, both rotating it.



Corn

Two brothers, one remote.



Herman

Quite. I'd expect some services to behave and some to throw you out, and I can't tell you which from here. One-way sync papers over it as long as only one side is active at a time. Which makes me want to ask you something, Corn. If it were your laptop, would you be comfortable with that extension sitting there pushing every login you own out the door, even to your own server?



Corn

For the sites that have no other way in, yes. For the bank, no. I'd rather the bank were annoying and slow than convenient and in a container.



Herman

Same. That's a line I'd draw in the config. Oh, and that's Mum on the line, put her on, put her on! I love this.

D

Dorothy

Hello dear, it is your mother, and it is time for my grim fact.

D

Dorothy

The peanuts. That is what they call the little hardcoded password Chromium uses on Linux to lock up your cookies, when there is no keyring to speak of.

D

Dorothy

Anyway. Have you eaten? I made a casserole and I am not going to eat all of it by myself.

D

Dorothy

Ring me when you are coming, sweetheart. Bye bye now.



Corn

Well, thank you, Dorothy. That was certainly a fact.



Herman

Back to Daniel's question, because a bigger threat than the debug port is coming, and it's called DBSC. Device Bound Session Credentials.



Corn

Which does what?



Herman

The browser makes a key pair inside the hardware. A TPM on Windows, the Secure Enclave on a Mac. The private key can't leave the machine. The site then only hands out short-lived session cookies, and each refresh requires the browser to prove it holds that key. So if an infostealer copies your cookie, it works for a few minutes, then it's dead weight, because the thief doesn't have the key.



Corn

And a copy streamed to a server is a copy the server can't prove anything with.



Herman

That's the logic. It's public on Windows from Chrome one forty-six. Google's post says the Mac version is coming in a later release. Okta and Microsoft were named as taking part. Google says nothing about Linux, and says it's exploring software keys for machines without secure hardware.



Corn

Daniel's on Linux.



Herman

He is, and I wouldn't build a theory on what Google didn't write. But if there's no hardware-bound path for his machine, the cookies he'd copy don't have that property. It's also opt-in for sites. Nobody has listed a Google account or any specific big site as using it. One report says the rollout went default-on in late May, and I can't confirm that anywhere.



Corn

So DBSC is the thing that kills the stream, for people on other operating systems, for sites that adopt it, eventually.



Herman

Yes. And that's good for the world and awkward for this exact trick. Which, funnily, makes Daniel's Linux laptop the last place it works.



Corn

Slow the whole thing down, which is my favourite speed. What happens when the server's browser shows up with the laptop's cookies from a different machine? Do the sites not notice?



Herman

Some do. Sites tie sessions to things like the address you come from and a rough fingerprint. But they're on the same network at home, and I'd say they probably share one public address behind the router. That's an inference, I haven't seen his network, but it's how a house normally works. It means the session doesn't suddenly appear from another country.



Corn

So at least the server isn't phoning in from a data centre in Frankfurt.



Herman

Not from Frankfurt, no. Now, the second design, and I think it's the one Daniel should take seriously. Skip the stream. Put a persistent profile on the server, with its own data directory, and log in once by hand. You run a browser in a container with a remote desktop in a web page, noVNC or the one called neko, you open it from the laptop, and you type your passwords and your two-factor codes into it like a person. Then you close the tab and the profile stays logged in.



Corn

No VM.



Herman

No VM. It's a container with a window. The cost is that when a site expires the login, you walk back to the window and do it again. Maybe every few weeks for some, never for others.



Corn

That's what I'd choose first, honestly. It's boring, it's repairable, and nothing is copying anything. I distrust anything with an event stream.



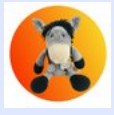
Herman

You're going to say you're indifferent about it and then win the argument.



Corn

I'm indifferent. I'm also correct.



Herman

Fine, but here's the catch. Some sites log you out the moment they see a new device, and a few will challenge you on the first login from the server's browser. You'd probably do that once. The other catch is the sites with short sessions, where the hand login is a weekly chore, and that's the one place a stream earns its keep.



Corn

And the third option is leave the browser exactly where it is and make it eat less.



Herman

Which is the least glamorous answer and might be the most useful. Chrome's Memory Saver, under Settings, Performance, parks tabs you haven't touched and reloads them when you come back. There's also a page that shows every tab and whether it can be discarded. I'm hazy on whether you can exempt a particular site from discarding, because the guides I've seen disagree.



Corn

The agent's tabs are the ones that get reloaded under it, mid-task.



Herman

That is the risk. A discarded tab loses its state, so a half-filled form comes back empty. I'd turn it on and watch whether the agent trips over it.



Corn

Mine would trip over it once and then lie down.



Herman

Fair. So what would you tell Daniel to do? You've been turning it over.



Corn

Layer it. Memory Saver on, today, costs nothing. Then the profile on the server, logged in once through a window, for everything with long-lived sessions. And for the handful that expire quickly, the narrow stream, three or four hosts, over Tailscale, encrypted at rest. Laptop Chrome stays for anything fussy.



Herman

The layering I like. The part I'd argue with is the laptop staying for anything fussy, because that quietly keeps the memory problem Daniel started with.



Corn

It keeps a smaller version of it. The agent's pile of tabs moves to the server, and the laptop only carries what he's looking at himself. Most of the weight was the agent anyway.



Herman

That I'll grant. And the server's logins would want a visible heartbeat, a page or a message that says which sites are still signed in. Otherwise a dead login looks exactly like a slow website.



Corn

A health page for cookies. The least exciting dashboard in the house, and the one I'd check first.



Herman

Okay, so say the stream's in. Say it does break, Corn. How long would you give it?



Corn

Three in the morning, some night in the first month. And nobody finds out until the agent's halfway through something. Who'd even notice?



Hilbert

Nobody, Corn. Forty-one pounds for a year of a parcel tracker, and it logged me out in week three and charged me again in week five. So on Thursday the browser goes into Gerhard Voss's shed.



Herman

A shed would be on a different network, which means a different public address from the laptop. That's the thing we said was probably shared at Daniel's.



Hilbert

Same address. His cable and mine come off the same pole. He runs it through the wall to the shed for Brenda. She's an alpaca. She's his sister-in-law's, and the sister-in-law is in Aberdeen.



Corn

And Gerhard lets you plug in.



Hilbert

One socket and the broadband. In return I read her the parcel status at six every morning. She only settles if someone reads out a delivery. So I have to keep one in transit at all times, and I order the cheapest thing on the site every nine days. She was posted once, to the wrong address, in June. She's technically still being tracked. Herman, small thing. An update isn't one event. I logged it at the library. Every login gives you two lines. The first one says removed. Then the second one writes it back.



Herman

Two, yes. A removal with the overwrite reason, then an explicit write. I said one, and that's wrong. Replay them in order and the server's logged out for a moment every time the site refreshes the cookie.



Hilbert

That was my forty-one pounds. Twice.



Corn

So the stream needs to throw away the first half and keep the second.



Hilbert

Gerhard ate eleven pickled eggs in six minutes at a ploughing match. Four of them out of the jar with his hand. The man next to him did ten, and Gerhard still brings it up in front of the alpaca.



Corn

Gerhard's a competitor.



Hilbert

He's a liar. It was ten and a half. My brother-in-law drove the van that lost her in June, and he says the same. Thursday I bring the cable and I log in once, by hand, in the shed, with the alpaca watching.



Corn

Ten and a half isn't a count.



Hilbert

She's better at six o'clock than Gerhard ever was, and he knows it.



Herman

Whatever the transport, the first rule is that the stream drops the removal that comes from an overwrite. It's in the cookie documentation, there's a reason field on each event, and the cause tells you whether it was an overwrite, an expiry, or a user logging out. You only act on a true removal.



Corn

Which is the difference between a stream and a screaming match.



Herman

And you only know that if you test one login end to end. Another thing to test is whether the extension actually sees the protected cookies, the ones the page's own scripts can't read. The flag exists on the cookie. I'd expect it comes back, since the extension API is meant to be more privileged than a web page, but I haven't seen it spelled out, so try it before you build anything.



Corn

Which is the cheapest part of the experiment. An afternoon.



Herman

It is. An afternoon with a throwaway site, a container for the server, and a stream with three events logged. If the second event shows up where we said it would, you've learned more than a week of reading.



Corn

Then the answer to Daniel is that it isn't impossible, and the side door is really the only door left. The debug port is closed, the extension route is open, and the hardware binding is coming for people who aren't him. For him, I'd start with the window on the server and see how long a login lasts before touching the stream.



Herman

And I'd keep the stream narrow, on a short list, over a private network, with the whole thing encrypted at rest. If a site quietly adopts hardware binding one day, the logins just stop working, and that's your cue to go back and do it by hand. That day might come late for Linux, though that's me guessing.



Corn

The thing I'd watch is the removal event. The whole design lives or dies on whether the stream knows a real logout from a cookie being refreshed.



Herman

Thanks to Hilbert Flumingtop for producing.



Corn

This has been My Weird Prompts. If you want to hear more, the website is my weird prompts dot com.



Herman

See you tomorrow.