Episode Book

EPISODE #6181

Shipping Private Android Apps to Your Family

PUBLISHED	21 October 2026 at 15:11 UTC
DURATION	22:22
LISTEN	myweirdprompts.com/episode/private-fdroid-repo-family-apps-90147/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.11
BOOK GENERATED	22 October 2026 at 07:46 UTC

SYNOPSIS

Building an app with AI is now an afternoon's work — getting it onto your aunt's phone is the hard part. This episode walks through running a private F-Droid repository from GitHub Actions: signed APKs, a signed index, HTTP Basic Auth, and a QR code treated like a house key. Along the way: why GitHub Pages can't host it, why Cloudflare Access would probably break the client, what Google's new developer verification actually covers (and what's still uncertain), and the two keystores you must back up or lose forever.

TRANSCRIPT



Corn

Build an app in an afternoon, then spend the evening working out how to get it onto your mother's phone. That's the situation Daniel is in. He's writing small Android tools with AI doing most of the typing, they'll never go near a public store, and the default is to build an APK, send it over, and hope. Updating is worse. So he wants the convenient way to ship and keep updating a private app for himself and a few non-technical relatives, and he's been circling a private F-Droid repo, run from a GitHub Actions workflow, with signed APKs, nobody else in it, and no way for a stranger to find it.



Herman

And he's right that it can work. We touched on the mechanics months ago, the fdroidserver tool plus a static web server. What's different now is the privacy half, what the F-Droid client will actually do about logins, and a cloud hanging over all sideloading. Google's developer verification went live on the thirtieth of September, a Tuesday, in Brazil, Indonesia, Singapore and Thailand.



Corn

Four countries. Jerusalem doesn't make the list.



Herman

Not yet, and the scope is narrower than the headlines. Google's own FAQ says the requirement applies to installs from participating stores, Google Play, Samsung's store, a few Chinese manufacturers' stores. For other stores and plain sideloads it says the requirements don't apply yet. Some press coverage reads it as already reaching unregistered apps. I can't square the two, so I'd call that uncertain. Global rollout is meant to come in twenty twenty-seven, with no date.



Corn

So it's not a problem today, and it's a problem on a calendar.



Herman

The friction is what matters. The escape hatch for unregistered apps is something Google calls the advanced flow. You enable developer mode, go through a coercion check, reboot, wait twenty-four hours, confirm with a biometric, and only then allow unverified installs. You can pick seven days or indefinitely, and you still get an Install Anyway warning.



Corn

Daniel can do that. His aunt cannot do that. Nobody's aunt waits twenty-four hours for a calculator app.



Herman

Daniel's own phone is the smaller problem, since Google says plainly that a developer can install over the debug bridge without verification. The aunt's phone is the real one, which is why the free route matters. There's a limited distribution account, no government ID, up to twenty devices, aimed at teachers, students and hobbyists. A household fits inside twenty with room left over. I haven't confirmed how you enrol the package name or the devices, so I'm not going to pretend I know the clicks.



Corn

Fine. Park that, and go back to the thing he actually asked. Why F-Droid at all? What does it fix that a pile of APKs in a chat thread doesn't?



Herman

Updates, mostly. A repo is a folder of APKs plus a signed index. The client on the phone fetches the index, sees a higher version number, downloads, installs. The family installs F-Droid Basic once, adds your repo once, and after that it behaves like a store. As far as I know, on Android twelve and up it can update without the privileged extension, so updates mostly happen quietly. And Android fourteen added update ownership, where the app that first installed something gets to update it without asking again.



Corn

Mostly quietly. That's the sort of sentence that ages badly.



Herman

Fair. I'd test it on the actual phones.



Corn

So the repo. fdroidserver builds it, you put it on a web server, and the web server is the whole question. Public, it's a repo anyone can add. Daniel wants it his alone.



Herman

Here's the part I find surprisingly good. The F-Droid client supports HTTP Basic Auth. Since the interface overhaul in twenty twenty-four there's no login prompt any more, so you put the credentials in the repo address itself, username and password, then the host name. The client sends them with every request.



Corn

Password in a URL. That sounds like what you'd get from a very enthusiastic intern.



Herman

It does, and that's why HTTPS isn't optional here. With a plain connection the password goes across in the clear on every single fetch. Over TLS, it's a bearer secret, and the repo address becomes the secret. If you share it as a QR code, the QR code is the key to the door. Someone photographs it over her shoulder and they're in.



Corn

How bad is that? It's a family tool. The worst case is a cousin installing my fitness tracker.



Herman

For apps that hold real data, worse. But I'd think of it in tiers. Tier one, unlisted: a long random path on a public host. Nothing links to it, but anyone with the address can read it. Tier two, Basic Auth: the server refuses everyone without the password, and the index and the APKs both sit behind it. I'd want tier two. Corn, would you bother with tier one? You like the slow, simple option.



Corn

For a calculator, yes. For anything with my data in it, no. Unguessable is a hope, not a lock. I'd want the server to say no.



Herman

Same. And the GitHub shortcut doesn't give you that. GitHub Pages is public even when the repo is private, on most plans, I'm saying that from memory. The popular template for running an F-Droid repo from GitHub serves it straight off the raw content address of a public repo. Raw links for a private repo need a token, and the F-Droid client can't attach one. So Actions builds it, but it can't be where it lives.



Corn

Then where does it live? His own server?



Herman

Your own server behind Caddy or nginx with Basic Auth is the plain answer. A bucket fronted by a small worker that checks the header would also work, but that's my sketch, not something I've seen documented. And one thing to leave out: Cloudflare Access with a login page. Service tokens work through custom headers, and I've found nothing saying the F-Droid client, or the alternative clients like Droid-ify, can send custom headers. A login redirect would very likely break them. Untested, so treat that as a warning, not a fact.



Corn

Hm. And the hostname? He thinks it's non-discoverable because nobody knows the name.



Herman

The moment you get a certificate for a hostname it can show up in the public certificate transparency logs. A wildcard certificate keeps the exact name out. Daniel's own radio repo already shows how this goes, it's public by design at a plain, guessable address. A private one wants the opposite.



Corn

So a long, boring hostname buys me nothing and a short one gives me away. Noted.



Herman

One more design choice on the login. A single shared password for the whole family works, but then a lost phone means changing it on every phone. If the server holds one username per person, you can cut off one relative without touching the others. It's a few extra lines in the server config, and I think it's worth it even for five people.



Corn

Five people, five passwords, and I'm the help desk for all of them. I'd still do it. A phone in a taxi is a better bet than a phone in a safe.



Herman

There's also the question of what the index reveals. It lists every package name and version you publish. Behind the login nobody sees it. On a public host, anyone who found the address would learn what your family's apps are called, and I'd rather they didn't.



Corn

So privacy is the server saying no, a wildcard cert, one login each, and a QR code treated like a house key. Now the Actions part. Where does the signing happen, because that's the bit that makes me nervous.



Herman

There are two keys and people blur them. The repo key signs the index. It's what the client pins when you add the repo. You can put its fingerprint on the end of the add-repo link so the client pins it on first contact, though I haven't read the exact client behaviour in a primary document. Then each app has its own APK signing key, and Android enforces that on every update.



Corn

Two keys. Lose which one, and what happens?



Herman

Lose the APK key and every phone has to uninstall the app before it can take a build signed with a new one. Uninstalling wipes the app's data. Lose the repo key and the clients stop trusting the index, so everyone re-adds the repo. Neither is catastrophic with five relatives, but both are annoying, and the first one is permanent.



Corn

So it's not one backup, it's one backup each, somewhere that isn't the thing that's compromised.



Herman

Yes, and that's the tension with Actions. F-Droid has no official Actions support. The usual pattern is to encode the keystore as base sixty-four, store it as a repository secret, decode it in the workflow, and let fdroidserver read the passwords from environment variables. Secrets have a size cap, around forty-eight kilobytes I believe, and a keystore fits. The workflow builds, signs, runs the update step, and syncs the output to your server over SSH.



Corn

Walk me through one release, start to finish, in words I could say to the aunt.



Herman

You push a tag. The workflow wakes up on GitHub's machine, builds the APK, signs it with the app key, copies it into the repo folder, runs the update step that regenerates and signs the index, and then syncs that folder to the server. The server only ever receives finished files. It never holds a key. One practical trap: the version code inside the app has to go up on every release, or the client looks at the index, sees the same number, and decides there's nothing new.



Corn

Forty minutes of debugging, then, to learn you forgot to add one.



Herman

Roughly that, yes. Easy to automate away, too, by deriving the number from the run.



Corn

So the keys sit in GitHub.



Herman

In GitHub's secret store, decrypted only while the job runs. The risk is anything that can edit the workflow can print the secret. A private repo, no pull requests from strangers, third-party actions pinned to a commit rather than a tag, and an environment with a required approval for the signing job. I'd also keep a second copy of both keystores offline, in a password manager. Do not let the only copy live in an Actions secret, because you can't read a secret back out.



Corn

You can't read it back. So the secret is a one-way door. Brilliant.



Herman

The F-Droid docs do say the repo key shouldn't ever sit on a public server, and they support smartcards and hardware modules. For a family repo that's overkill, but it's good to know the ceiling exists.



Corn

Herman, brace yourself. And now it is time for our next segment, Dorothy's Kitchen.

D

Dorothy

Hello dear, it is your mother. I am here with Dorothy's Kitchen. Today we are making a cracker with nothing on it.



Herman

Mum! Hello! What a lovely surprise.

D

Dorothy

You take one cracker. That is the ingredient. That is all. It sounds simple but it is not.



Corn

Just the one cracker, Mum?

D

Dorothy

First you watch the cracker. You watch it very closely. Do not look away, sweetheart. The oven can sense fear.



Herman

How closely, Mum? Like a hawk?

D

Dorothy

Then you wait. Two minutes, or twenty, it hardly matters now.



Corn

Is that with the oven on?

D

Dorothy

We do not speak of the incident of nineteen ninety four. I will only say that the smoke alarm was involved and your father was not helpful.



Herman

Understood, Mum. We will take notes.

D

Dorothy

Anyway. Eat it over the sink. Wear a cardigan if you are going out, and give my love to your brother, even the donkey one.



Corn

Bye, Mum.



Herman

Back to the keystore, the one you can't read back out. Corn, when a relative gets this on her phone, what does she actually have to do? You're the one who gets the phone calls.



Corn

Nothing, ideally. One install, one scan, and then she never thinks about it again. If there's a permission screen, she calls me.



Herman

That matches how it goes. One detail on the client: the F-Droid Basic build is the stripped-down one, and the newest I've found is a two point oh release candidate from August. I haven't seen a final, so check what's current. Android asks once, per installer app, whether it may install unknown apps. You hand them a phone with F-Droid Basic installed, you grant that permission, you scan the QR with the credentials in it, and you're done. After that she gets a notification or, ideally, nothing at all. I would set it up for them on a visit, in person, because the first run is where a non-technical person gets stuck.



Corn

And if the notification says an update is waiting and she ignores it for four months?



Herman

Then she runs the old version for four months. That's the limit of any scheme that doesn't force updates. On to the alternatives. The one that fits a technical user is Obtainium. It installs and updates from release pages, GitHub among them, and you can share a link that pre-fills the app configuration. For a private repo you create a fine-grained personal access token, give it read access to the contents of that single repo, and paste it into the GitHub source settings. That route is community-reported rather than something the documentation spells out.



Corn

So each family phone holds a token that can read the repo.



Herman

Yes, and fine-grained tokens have to expire, so they lapse. Put a token inside a shared config and whoever gets the config gets read access to the repo. And because the token belongs to your account, the requests are linkable to you. For my own phone I'd do it. Corn, would you hand a relative something that expires on a schedule?



Corn

I'd hand her a phone that never asks. Then no. What about the proper stores? Play must have something for people who don't want a public listing.



Herman

Play internal testing. Up to one hundred testers added by email, the build is available within minutes, and testers opt in through a link. Updates come through Play, so it's the best experience of anything on the list. Costs: the one-off twenty-five dollar developer account, Google's verification, and Google's own page notes that internal builds might not go through the standard policy review. And you're then in Google's hands.



Corn

A hundred is more relatives than I have, and I have a lot of relatives.



Herman

Closed testing goes to two thousand, but a new personal account has to run a closed test with twelve testers for fourteen days before it can go to production. That's irrelevant if you stay in internal testing and never publish. Then there's Firebase App Distribution, up to five hundred testers, invites by email or link, with an option to restrict acceptance to the invited Google account. Every tester needs a Google account, though, and the setup is meant for developers testing a product, not a household.



Corn

And the privacy-minded one, Accrescent?



Herman

Rules itself out. It says plainly it has no plans for third-party repositories, so there's nowhere to put your own apps. Managed Google Play private apps I haven't looked into. It needs a workspace and device management, so I'd bet it's overkill.



Hilbert

...and that's why the biscuit tin had to stay at thirty-seven degrees. Mr Threlfall tested it with his wrist. The goose sat on it from Monday and was only relieved on Thursday, so you had a Thursday run and nothing else.



Herman

Forty-one members is a small enough fleet to do by hand, at least.



Hilbert

The Kilburn and District Disk Exchange Society, forty-one members, one rule book. Rule fourteen said the post was not to be used. So the van went out. That was me, in the van, for Pettigrew Micro Supplies, with the updates in the tin. Corn, you're peaking on the S's,



Hilbert (continued)

I'll bring it down. And nineteen eighty-eight is when they voted on whether the van could exceed nineteen miles an hour, because of the shaking.



Herman

Nineteen miles an hour is a long update window. At forty-one members you're out most of a day.



Hilbert

Nine hours and ten minutes. The thirtieth was a Wednesday, by the way, Herman. You said Tuesday. It doesn't matter. The committee minuted the speed limit as a Wednesday decision too, in the end, although they'd argued about it on a Thursday.



Corn

So the goose is on the payroll, and the van is on a leash.



Hilbert

The goose was a member. Fully paid. Treasurer for a while, though Mr Threlfall kept the chequebook, because of the beak. I did all the stamps myself. I used my tongue. Pencils don't suit my claws, and the glue was a lemon glue, from a firm in Leeds that has since stopped.



Corn

Then the dispute is the delivery day.



Hilbert

Eleven years. Tuesday or Thursday. The Tuesday faction, there were six of them, called themselves the Early Disk. Minutes ran to four hundred pages. The tin I know exactly where it is. Raymond has it. My brother-in-law, in Rotherham, on a shelf, with a sock in it.



Herman

And the disks went in the tin because of the warmth.



Hilbert

Mr Threlfall believed it. Nobody was going to tell him. Herman, I've gone and looked at where I keep the key to my own garage, after what you said about the signing key.



Corn

Hm.



Hilbert

Mrs Garrity never got hers. She lived at the top of a hill the van wasn't allowed up. She ran the old version for six more years and said it was fine.



Herman

Six years on the old build. That's the number I'd plan around. The person who's furthest from the update is the one whose phone you should test on.



Corn

And that's who gets the scan, before anyone else. If it works for the aunt who calls me about the volume button, it works for everyone.



Herman

So here's what I'd set up for Daniel. A private F-Droid repo, built by a private Actions workflow with the signing keys as secrets and offline copies of both keystores. The output goes to your own server over SSH, behind HTTPS and Basic Auth, on a wildcard certificate so the hostname isn't in the logs. The family gets F-Droid Basic and one QR code, installed in person. That's the default.



Corn

And I'd add the free Google limited distribution registration before twenty twenty-seven, if it turns out to cover this. Twenty devices, no ID. If the advanced flow ever lands on direct installs, that's what keeps the aunt from the twenty-four hour wait.



Herman

When would you not pick that?



Corn

When a relative's phone is managed, or locked, or just too far gone to set up. Then it's Play internal testing. Twenty-five dollars, a hundred people, Google's problem. And for your own phone, Obtainium with the token, since you're the only one who'll remember to renew it.



Herman

I'd agree, and I'd do the least technical person's phone first. Before Daniel builds all of it, though, there are three things I'd test in an evening rather than trust me on. Whether the F-Droid client on a current phone really accepts the login in the address. Whether an update lands silently on an older Android. And whether the fingerprint on the add link behaves the way I described.



Corn

Three tests, one spare phone, one evening. That's cheaper than the twenty-five dollars, and you can't do it wrong in front of your family.



Herman

One last practical point on the Actions side. Set a calendar reminder for anything that expires, a token, a certificate, a password you rotated. Quiet failures are what a private repo is good at. Nobody notices until someone's phone has been three versions behind for a month.



Corn

Which is a good argument for a repo I check on, not one I admire. Next time we do the same question for Linux, private repos for packages, which means APT and what sits next to it. Thanks as ever to our producer Hilbert Flumingtop for keeping the desk running.



Herman

This has been My Weird Prompts. If you want more, the website is my weird prompts dot com. We'll be back soon.



Corn

See you tomorrow.