EPISODE 1516

[INTRODUCTION]

[00:00:00] ANNOUNCERP: This episode is hosted by Sean Falconer. Sean's been an academic founder and Googler. He has published works covering a wide range of topics from information visualization to quantum computing. Currently, Sean is Head of Developer Relations and Product Marketing at Skyflow, and host of the podcast Partiality Redacted, a podcast about privacy and security engineering.

Passwordless authentication is a technique in which users are given access to an environment without entering a password or answering a security question. This allows users to access an environment securely, and protects organizations against attack vectors like keylogging, brute force methods and phishing.

The company SuperTokens provides secure login and session management for your apps and an open core model. In this episode, we interviewed Advait Ruia and Rishabh Poddar from SuperTokens. We discussed open source authentication, security considerations for authentication, recipes for authentication, and the future of passwords.

[INTERVIEW]

[00:01:07] SF: Adi, Rashad, welcome to the show.

[00:01:09] AR: Yeah. Thanks for having us.

[00:01:11] RP: Nice to meet you. Thanks for having us.

[00:01:13] SF: Yeah, nice to meet you as well. This is both of your first time on the show. I think it's a great place to kind of start with some basics and have you introduce yourselves. Who are you and what is it that you do at SuperTokens? And maybe, Adi, you can start?

[00:01:27] AR: Yeah. I'm the CEO and cofounder of SuperTokens. And Rishabh is my cofounder and CTO as well. In terms of a little bit more about us, we've been working in building startups for the last about six to seven years. SuperTokens is a dev tool company. We do user authentication.

[00:01:42] SF: Great. Rishabh, do you want to introduce yourself as well?

[00:01:46] RP: As Advait said, I'm the CTO and cofounder. Essentially, been programming since 15 years now. And a little web programming. Dabbled in several, several frameworks on the frontend, backend. That's a lot of useful information to build an authentication solution. Yeah, worked at various startups over the years. And now, SuperTokens.

[00:02:06] SF: Fantastic. And obviously, we're going to dive deep into SuperTokens today. But before we get there, authentication and authorization is something that has come up on the show in the past. It's something that comes up in a variety of different formats. I recently spoke to the founder of Permit.io, and they essentially have an authorization product. But despite talking about these topics all the time, even I sometimes need to pause for a second and remind myself which is which. I think they mistakenly name these things very, very similar. But maybe we start there. What's the difference between authentication and authorization?

[00:02:39] AR: Yeah. Authentication is knowing who is the user. And authorization is knowing what do they have access to. If you like take a physical example. So, authentication is knowing, let's say, you walk into a building, and then knowing that, “Okay, this is Sean that's entering the building.” And then that's authentication. And authorization is then saying, “Okay, he can access these rooms, these floors, these cabinets.” And that’s sort of all authorization.

[00:03:02] SF: Mm-hmm. Yeah. That's great. And then I imagine all sorts of modern systems, essentially, you're using a combination of these two things to both understand who the user is, and then also understand what they actually have access to.

[00:03:13] AR: Yeah. You always need authentication. And then depending on the app – I mean, a lot of apps do. But not all apps require authorization. But anything with users definitely requires authentication first, right?

[00:03:23] SF: Right. Back in the day, not really that long ago, everyone kind of did their own authentication. But there's been this industry trend to kind of move away from that. I guess, what makes authentication hard? Why not just continue to roll your own authentication?

[00:03:39] RP: Right. First, it doesn't always have to be difficult. If you have a hobbyist project, or a very simple project not used by many users, and you just want, for example, email, password-based login, then you can roll out your own in a few hours, and it will work just fine. Right? You don't need to worry about anything else. But as soon as you're building a more serious app, like for a startup or within an enterprise that you expect to be – for b2b apps, you expect to have high value customers. Or for b2c apps, you expect to go to millions of users using application, that's when things start to get a little bit difficult.

And let me sort of elaborate why. For b2b apps, where each customer is of high value, you want to make sure that the session or the account is not taken over. And that even for something like email, password login, which hashing algorithm to use? For example, you would want to go with the more secure one, which is, for example, Argon2 hashing.

But when you do that, and if you sort of just google, “How do I add Argon2 hashing to my app?” It will give you code snippets for how to do that. But it won't tell you that Argon2 is a memory-intensive algorithm. And if you don't have any rate limiting on top of Argon2 hashing, somebody can just spam your sign and form and sort of crash your servers because there's no more RAM left. That will be mentioned in those blog posts, right? Or even if it’s, it won't tell you how to do those rate limiting. Things like this is what makes something as simple as email password difficult.

Another example, from the user experience side, and this would be useful for consumer-based apps, is if you have like magic link-based login, you would expect that when a user clicks on the link from the email, they will get logged in. But you have to consider the edge case that what if the email scans the link for viruses, which, for example Outlook clients do? Then they would end up consuming the code in the token. And when the user clicks the link, they will not be able to log in, right? And that obviously leads to terrible user experience. And it's an edge case that you have to design for.

When it comes to things like session management as well. For b2c apps, you want the sessions to be long-lived. You want to scale to millions and hundreds of thousands of users. In that case, you want to go with something like JWTs, because they are very easily scalable. But what if your JW signing key is compromised? Then, essentially, the attacker can assume the identity of any account on a system. And that's obviously very bad. How do you prevent against that? How do you build a system which automatically rotates the key without logging anyone out?

For b2b apps, you want to make sure you follow all the checkboxes when it comes to security of session. HTTP only cookies, CSRF prevention, all of the things that can prevent session hijacking. When you think about all of these things, then then you realize that, okay, you need to be very knowledgeable about security to be able to actually implement these for a production system.

And so, you can hire like a security expert for these things, or a team of experts to implement this. But then what happens when they leave your organization? And that's kind of the question about maintainability, right? You don't want to be in a situation where the person or the people who wrote your auth system have left and all your current engineers do not know how it works. And if you want to make changes to it, then it's going to be incredibly difficult and costly. These are the reasons why people nowadays tend to pick an authentication provider, because they've learned these lessons from experience of building their own that is all worth it.

[00:07:06] SF: Yeah, absolutely. I think you did a really, really great job of sort of articulating some of the, basically, death by a thousand cuts that someone kind of runs into as they start to try to build an authentication system for like a real application that's serving real users at scale.

It seems like, in a lot of ways, something like authentication is – it has like this iceberg effect, where 20% above the surface seems like pretty straightforward. You can kind of build that fairly quickly. But then there's this like long tail, which really represents 80% of the work, where you're dealing with all these potential security issues, things like you mentioned, the magic links and having Outlook scan those links and render the links invalid, and so on.

You mentioned a couple of the kind of like security concerns to think about when building an authentication system. Can you elaborate a little bit on that? And what other kind of security concerns would a company have to think through if they were actually building their own authentication?

[00:08:00] RP: Sure. This is huge. It's going to be very difficult for me to cover every single one of them in this podcast. But I'll go through like some of the most common ones that I've seen homegrown systems run into, like some of the issues.

Starting off with something like email, password login. You want to make sure your passwords are hashed and salted. And hashing essentially means that you're translating a plaintext password into an obfuscated form, such that if you have the plaintext, you can get to the hashed obfuscated form, but you can go back. And salting essentially adds variability to the output, so that common passwords are not detectable either.

And then another thing with passwords is like a very common way of leaking them has been API's logging passwords. Normally, when you have log alls for your API's, you tend to log all the request URL and the body and all of these things and sort of store them in S3 or somewhere. You want to explicitly not log users' passwords in that.

Another thing related to password is the reset password flow, where you want to be sure that the reset password tokens are stored in hash forms in the database, then they're not sent to the frontend. I've also seen that happen. And related to this is sort of the topic about secret management, which is your JW signing keys and your OAuth credentials. You want to make sure that you're using a key management system for these things that you're signing keys rotated from time to time. And then you have like, as I mentioned, that magic link issue.

And so, there's like a lot of things. The devil is already in the details about what you should really consider as good security practices. It really depends on the authentication method and the user experience that you want to build. There isn't like one answer that fits all here. It really depends on what you require.

[00:09:45] SF: I think that's like consistent when it comes to things like security and privacy. There's not one answer that fits everything, which is another reason why it doesn't make sense a lot of times for a business or a new company that's like building an application to build something like authentication themselves when they could be focusing their resources sort of on what the core product is. And what's going to deliver ROI. Because they don't want to have to kind of think through all these – like you said, the devil is in the details. Think through all these details that are not sort of core to who they are as a company.

And I like that you mentioned logs being a culprit for a lot of these issues. We see that all the time. A lot of sort of data breaches, data leaks are the result of – even if it's not passwords being in the log files, or some other value that's essentially personally identifiable information, that ends up in someone's log file.

[00:10:34] RP: Yeah. I mean, even if – Recently, GitHub Copilot, right? I mean, it's not recent. But they private repos on GitHub, they essentially had access to secrets of people, your GitHub credentials, and like your Google credentials, and all of those things. And then, if you did like an autocomplete on your VSCode, it would give those credentials as suggestions. That's a pretty bad leak as well. And that just goes to say that you have to be really careful about secret management as well.

[00:11:02] SF: Yeah, absolutely. I want to start to transition to talk a little bit about SuperTokens. What motivated you to start the company? And I guess how did it start?

[00:11:13] AR: Yeah. We were building this app – we were building our own app. And one of the things we noticed was that there were a lot of misconceptions around the concept of session management. The way tokens are stored. The way they're created. Even like, in our opinion, like there was a basic misunderstanding about using local storage versus HTTP-only cookies. And we saw like being pretty ubiquitous. We saw a lot of big companies that went through hacks. Even like Facebook, Docker, GitLab, YouTube, Uber, they've all had like some version of a session vulnerability.

We just basically device a pretty secure way of handling sessions ourselves. We build pretty extensive system. And then we wrote a blog post about how that system worked. We

implemented – we did this thing called rotating refresh tokens that allowed us to actually detect session theft, because there are only two ways of stealing a user's account. It's either through the login credentials or the session. And session vulnerabilities were prominent.

We devised that flow. We wrote a blog post. That blog post did really well. And that's sort of how it started. That's the initial exposure to it. And then we got into YC. This accelerator called Y Combinator. And then we like spoke to a lot, like hundreds of startups. Until then, we were only focused on session management. But at that time, when we start talking to a lot more people, we realized that the pain points were there even with the authentication space as a whole, not just with session management. And yeah, and then one thing led to another, and then we started doing everything we do today.

[00:12:29] SF: Yeah, amazing. I love the idea too, of focusing sort of on a problem that you recognize that's important to you. You solved that problem. And essentially, you realized from the – it sounds like from the traction that you got from your blog post, it was like, “Oh, wow! A lot of people are having this problem.” And maybe it's time to sort of productize this.

There're quite a few players in the authentication space, some big ones obviously like Auth0. What makes SuperTokens unique in comparison to some of these other companies?

[00:13:01] AR: Yeah. There are a few things here, just like the thing that's most prominent is the fact that we're open source. And being open source means it enables – It's good for developers in a couple of different ways, right? The obvious ones being the fact that you can self-host it. You can control and manage your own user data. It reduces the vendor lock-in. If tomorrow, they arbitrarily change pricing, you can always choose to opt out and self-host. You can always build on top of that core product, all of those kinds of things, right? There's obviously the open source side of things.

And then there's the entire architecture and the way we designed the product, right? Our architecture is fundamentally unique. And the reason that's important is because it gives you a lot more control as a developer. Or the application gives you a lot more control over the authentication system. For example, everyone does prebuilt UI. Every authentication provider gives you a UI that's prebuilt. But with most of the others data, that UI is hosted on an external

domain, like, the authentication provider's domain. Whereas with us, we give you the UI, and its native to your app, right? It's native to your own frontend. You have a lot more control over the way – it's unlimited control over your end user experience.

Similarly, the backend auth logic all resides within your API layer, right? As opposed to being on an authentication server somewhere. You can make those changes in Node, Golang, Python, which is your own – in whatever language your API's are written in. There's a lot more control and customizability in addition to being open source.

And then finally, we give you like an end to end session management system as well, right? Alluding to sort of our routes, right? We started off as that. We do that in the most secure way possible. We handle all the vulnerabilities. And again, because those are architected as third parties, we're architected as part of native to your app. We also handle like all the frontend, back end session management communication for you.

[00:14:41] SF: Is that a difficult thing to sort of balance between giving people control and also setting them up for success when it comes to sort of the security best practices that you should be leveraging within SuperTokens?

[00:14:55] AR: In general, yes, because there are always tradeoffs between control and making it simple to use, right? But in terms of security, obviously, the default is always the security best practices, right? We'll always only use HTTP-only cookies instead of local storage, right? The defaults will always have all of the things best – And very rarely would we ever even give you the option to not have a security best practice, unless there's a very good reason to do that.

[00:15:18] SF: I see. And then you mentioned that you're open source, and you also support, I believe, a SaaS-based solution to that. Essentially, someone can run this themselves using the open source project, or you can have like a SaaS-based hosted solution. What are some of the differences? I guess, like how do companies typically use SuperTokens? And what goes into that decision-making process of using the open source version versus using the SaaS-based solution?

[00:15:44] AR: Yeah, I'd say at this point in time, the majority of our usage. People do use both. But the majority of our usage is on the open source side. And I think the reason for that is like one of the things that makes SuperTokens unique today is the fact that it's open source, right? There're so many authentication providers. A lot, most of them are closed source. One of the things that makes us unique is that.

But yeah, and the decision making into that process is I think things that I alluded to earlier, which is self-hostable, control and manage your own user data, vendor lock-in, pricing. You already run multiple services. You're happy to just set up one more service. It's not a big pain. But you'd rather just have control over it. I think those are the reasons why you do self-hosted. SaaS is obviously easier to get started. One of the reasons you do SaaS is because it's easier to get started. And you just don't have to host the service.

[00:16:28] SF: Why was investing in open source like an important decision for the company? And I guess, how has that impacted your business and product offering as you've started to build out this company?

[00:16:43] AR: Yeah. Open source, in general, aligns with a lot of our values as individuals and things that we wanted to build. We want to be as transparent as possible. We want to keep the developer at the absolute center of our decision making, right? All the best companies are built on keeping the user at the center. And we think that open source is like – if you truly believe that, then open source should be the default choice, because it's just provides a better developer experience in so many different ways.

I think some of those reasons. And then, in terms of how is it more developer friendly, it reduces vendor lock-in. All the same reasons that we're discussing. And then there's also a sense of community. You can build a community with a closed source product. But we've just really enjoyed building one with an open source product in terms of how quick the feedback cycles and the feedback loops are. And then also, contributions. Now we've had over 200 contributions. People have made over 200 contributions to SuperTokens.

Things like that, I think. There is a business angle, of course, right? We also think it's a good way of executing a company, building a company. It's beneficial from a business point of view.

But even more organically in terms of just the decisions we want to make, this is just more natural to us.

[00:17:52] SF: Great. Yeah, that makes sense. And there's tremendous amount of value in open source just from getting your product in front of a lot of people to essentially accelerate a feedback cycle. Can you take me through the process of actually how to get started using SuperTokens? I'm going to build out authentication. I'm interested in using SuperTokens. What's the process to get started and kind of get up and running?

[00:18:16] RP: You start by visiting our website, supertokens.com. And when you click on the documentation link, it shows you a few guides, essentially, which we call recipes. Depending on what kind of authentication method you want for your app, you choose that guide.

For example, if you want passwordless login, you would click on that guide, and follow the quick setup section in there. And that will take you through the frontend and backend SDK setup, which is like a few steps in each case. And then you have to set up the call, which either you can self-host using Docker, or sign up on supertokens.com to get the managed version. And you're pretty much done. That's sort of the basic setup right there.

We also have specific guides for frameworks such as Next.js, and NestJS, and RedwoodJS, and all of these things, AWS Lambda, and all of that. You could also follow those in case you're using one of those frameworks.

Recently, we launched a CLI called Create SuperTokens App, inspired by Create React App. Essentially, what it does is it asks you a bunch of questions, like what is your backend written? So, Golang, Node.js, Python. What is your frontend? Is it Vue? Angular? React? And what kind of what authentication method you want? Email password, or passwordless, or a combination of these things. And based on your choices, it generates an application for you, which has all those choices put into it. And you can use that as a base to build your app as well.

[00:19:40] SF: Yeah. It sounds like it essentially is built through the CLI, building out the scaffolding for basic applications, right?

[00:19:47] RP: Yeah.

[00:19:48] SF: You mentioned these different recipes that you support things like passwordless, phones, social, user roles, and so on. What should developers be thinking about when choosing an authorization recipe?

[00:20:00] RP: We classify recipes into two forms. One is auth recipe and one is known auth recipe. Auth recipes are ones that create users. For example, passwordless, email password, or social login recipes. And non-auth are ones that take in a user ID. For example, the session recipe or the user roles recipe.

At a minimum, you would want to use one auth recipe along with a session recipe. For example, for an email password login, you would use the email password auth recipe and the session recipe. And then on top of that, you can add things like user roles, or user metadata, and machine to machine, all of these things based on what your requirements are.

In terms of the complexity and the security. So, each recipe is pretty independent of another. If you add more recipes, it doesn't really impact your experience when using one of them. The configs are pretty isolated, and clear, and defined. Scoped to that one recipe. The complexity doesn't really increase. The two things from a security point of view, one is like the security from a developer's point of view. How many things do they have to do to make sure the security of the app is fine? And one is the actual application security.

We obviously designed all the recipes keeping security best practices in mind. Regardless of the combination recipes you use, there's not much that you need to do there. But in deciding which recipe to use, you should think about UX as security. For example, if you choose an email password recipe, the attack vector somebody has to take over someone's account is the password as well as the end user’s email account. Whereas with passwordless, the attack vector just includes the end user’s email account when we're talking about magic link-based logon. That's something you have to decide on your own for your own app, like what you're okay with and what you're not. But other than that, we take care of everything else.

[00:21:50] SF: If I have already built an authentication system, so I've done like my homegrown authentication, how hard is it to replace it with SuperTokens? And what's that process like?

[00:22:01] RP: It's not as difficult as you might think. We've had people do it in under one day. But essentially, it requires two things. One is the data migration, and one is the code migration. When we're talking about data migration, the first step is user creation. For example, if you have email password credentials, you loop through all your users and call the signup API in SuperTokens with the email and password hashes. And that would create those users in SuperTokens.

For social login, you would do the same thing, but with the social login user ID and the user's email. Those would generate SuperTokens user's IDs for each of those users. Then you would go about mapping your existing user IDs to the SuperTokens user IDs. And that's also done via an API call to SuperTokens.

What this will do is if an existing user signs in with SuperTokens, it will give you back your existing user ID, which your existing application tables use. You don't have to change any of those. And then comes the step of if like your existing user's emails are verified, you want to also mark them verified in SuperTokens. And that's, again, one API call. You want to transfer the rules. And that's, again, like one API call. You want to transfer the user metadata, and that's another API call.

The data migration part is fairly straightforward. The core migration part is you want to follow the quick setup that we have for backend SDK integration. And for frontend, you can sort of – you can continue to use your custom – like, your own UI that you have. But instead of calling your API's, you would use our frontend SDK functions to talk to SuperTokens.

For example, when the user clicks on the sign-in button on the sign-in form, instead of calling the old sign-in API, you would just do like email password or sign-in from the SuperTokens frontend SDK. And that's about it.

What this means is that the end user doesn't really see the effect of the change. They don't have to reset the passwords. They don't have to go through human verification flow again. And

you can even make it so that they don't have to re-login. Current sessions keep working. It can be a pretty smooth experience from both developer and user point of view.

[00:24:05] SF: If I have an email and password set up today that I built, how does the, essentially, hash password that maybe I'm storing my database today get transferred over to SuperTokens? How does that sort of synchronization process or migration process work?

[00:24:19] RP: IT depends on the hash of the – the hashing algorithm that you use. We support several of them. Let's take an example of bcrypt hashing algorithm. We have an API where you can just – given him the email ID and the bcrypt hash, and it will store that as a user in SuperTokens. And then when you talk about bcrypt hash, the number of bcrypt rounds comes into the picture. You may have a different bcrypt rounds setting in your old auth system than compared to SuperTokens. But SuperTokens sort of takes care of that difference for you. When the user signs in, we will actually use the rounds configured in your old hash compared to the ones that SuperTokens has set up.

[00:25:01] SF: I see. And then it sounded like – so if I have an internal representation of a user today in my database, and then I start using SuperTokens, the mapping between my internal user ID and the user ID representation of SuperTokens is hosted by SuperTokens. Is that right?

[00:25:17] RP: That's correct. After you do the mapping, and if the user signs in, you get back your user ID from SuperTokens. And even when you do like session.getuserid, you get back your user ID from SuperTokens. There's not much you have to do on your end, other than just create the map.

[00:25:33] SF: I see. Yeah. That's really nice from a migration standpoint, because I don't have to introduce a new column into my user's table to maintain the SuperToken user ID or something like that.

How does something like testing work? If I'm doing integration testing through Selenium or Cypress and I want to actually test an experience that requires login, how do I go about setting something like that up?

[00:25:58] RP: There are a couple of ways. One is you could spin up a Docker image of SuperTokens core, which is not connected to a database. And that uses an in-memory database per test, or a per test set. And use that as the auth server for your test suit. You could use something like Puppeteer or any other tool that you mentioned. And when you call the login API, the login API would return session tokens, which again will be handled by our frontend SDK. And then for you, as a test writer, it would just be like writing a regular test, however you do it with your old system.

[00:26:37] SF: I see. You've mentioned some of the like frontend frameworks and technologies that you support. But what is the sort of extent of support on the frontend and backend for SuperTokens?

[00:26:48] RP: On the front end, we support all web frameworks. We have a dedicated React SDK. But we also have a Vanilla JS SDK. Anything you use on the web would work. For mobile, we support iOS, Android, React, Native and Flutter, which covers most of the technologies there. And we have support for Capacitor, Electron, Cordova, the hybrid frontend applications as well. But you have to make a few customizations to how the frontend SDK works to get the support to get working in those environments. But we have example apps for those.

On the backend, we have SDKs for Node JS, Golang, and Python. We integrate with all the web frameworks within these languages. And that being said, if you use something other than these three, you can always spin up a service, which is running in one of these languages, and then use that as your auth server, which issue the JWT. And that gets consumed by your application backend.

[00:27:41] SF: Right. Are these all built internally at SuperTokens and then released as open source? Or are some of these community-built libraries?

[00:27:50] RP: At the moment, all of them are built by us internally. We have a community-built C# SDK that focuses just on the session recipe. But until we actually do that, until we actually come and build a C# sharp thing from ourselves, I guess it's up to the community to work on that.

[00:28:07] SF: What goes into your sort of decision-making process about which languages and frameworks to support and kind of like where you dedicate your resources and prioritize these different languages and technologies?

[00:28:20] RP: Most of us it is to do with the popularity of the framework. On the frontend side, at this point, we support mostly everything. There's not much more left to do in there. But on the backend, we have loads of things like C#, and PHP, and all of these things. But the reason we picked Node.js, Golang, Python is just because one is we were comfortable with these languages ourselves. And also, we felt, for startups, which is sort of our current target market, these were the popular languages that they used.

[00:28:50] SF: You mentioned startups. Is that your typical customer? Or are there also sort of larger enterprises that are using you as well? Or is there just essentially a range of different types of customers?

[00:29:01] AR: Yeah. The bigger the customer is more – The things we focus on currently are more the startups. Everything from like developers building side projects, to early stage startups, to series B companies. That's the typical customer profile. But having said that, we do have larger companies. We do have companies that have raised like 100 plus million, or that are worth maybe a unicorn and things like that. We do have those companies using us as well. But the typical customer is, yeah, younger.

[00:29:26] SF: I see. And we talked a little bit earlier about some of your decision-making around open source and also the value that being in the open source community has given to SuperTokens in terms of generating feedback and so on. But you also have things like an ambassador program. And how is the community program for something like the ambassador program benefited or positively impacted your business?

[00:29:53] AR: Yeah. The ambassador program was an interesting experiment. We did that in the early days. We wanted to sort of see like how it's like. It ends up being obviously junior. Like, people more on the junior side. And that's interesting for multiple reasons, right? Obviously, these are people looking to learn. It helps us understand what are the gaps in what

SuperTokens – how we're communicating and, how we're educating users, and things like that, right? And I think it worked pretty well. We did have a growth.

Those are one of the first programs we ever did, or like anything that we've done at all in the distribution side. It was interesting. I mean, even today, people reach out regarding that program all the time. But that was like one of the community programs we did. I think we've done a bunch of others. We integrate with, for example, other dev tool companies, right? Like you mentioned permit.io earlier, the one company we've worked with in the past. And there are multiple others where we do sort of like an integration. And that sort of becomes the community program where we all – where we leverage off each of those sorts of communities and get everyone involved. That's also worked well.

And then, yeah, in general, again, we were very active on Discord, right? Our native community is on Discord. People who are interested in SuperTokens joined Discord. And we're pretty – We're very responsive in terms of support questions and things like that there. Yeah. And then we try and be transparent. We try and create as many of our issues and as many of our like things on GitHub, right? So, people can track it. People can create issues. People can comment. We create as much as possible. We're as transparent as possible about product roadmap. And like we have a product roadmap page, right? Which lists out all the different features we're working on. All of these are like also part of community. They're not directly like a specific program, but they're all like things that help aiding in that community.

[00:31:34] SF: Yeah, for sure. I think to do community right, it kind of has to be part of, baked into the culture of the company. It goes beyond just single program. Or it's not a feature that you just flip on and do a two week sprint on and things like that.

[00:31:46] AR: Exactly, exactly.

[00:31:49] SF: You talked a little bit about your founder journey at the beginning. And you, I think, painted kind of a very rosy picture of what you've been able to do, where you wrote a blog post, you entered YC as a result, you raised some money. And now you're off to the races, which, as an ex-founder myself, I'm sure that the actual picture is a lot more difficult than what

you had shared. But as founders, what is surprised you the most about sort of building a company and the product that perhaps you didn't expect when you started down this journey?

[00:32:18] AR: Yeah. This is code, right? The journey of a thousand miles begins with someone who doesn't know how long a thousand miles is, or something to that effect, right? Actually, it's a code by the cofounder of Okta, right? By Todd. I think that's like everything that that encapsulates a lot of things that are surprising, right? It's like, for the most founders, it's going to be a lot harder and longer than you would have expected. You always expected to be hard, right? It's not going to be easy. But you always expect, "Oh, yeah, in two years, I'll build this massive company. And in one year, I'll figure everything out. In two years, I'll be scaling it. And five years, I'll be this is – It'll be like the default that everyone uses. And that's sometimes the case. But more often than not, that's not the case, right?

I think like learning all of that is, it's not the most surprising, but it's also the most unexpected, right? Because you never expect that to happen to you. Because you always expect to be the exception. Yeah, I think that's been like interesting. And we've learned a lot from that. Just the number of intellectual, emotional and physical challenges that that comes with, right? Physical in terms of the number of hours, intellectual in terms of like the difficulty of the problems that you're solving. And everything is uncertain. You don't really have a clear answer to many of these intellectual problems. And that lends itself to the emotional side of it. So yeah, there're multiple things to learn from all of that.

[00:33:33] SF: Yeah, I think it's great. The one that I used to like when I was a founder was that the don't mistake a clear path for a short one. Just because you know the things that you should be doing doesn't mean it's going to happen overnight. It takes a really long time to actually get there.

Zooming out a little bit to kind of just thinking about the future of authentication, do you think are passwords ever going to go away? This is something that's been around for 50 plus years? And are they going to go away? And what alternatives do we kind of have looking to the future of authentication?

[00:34:06] AR: Yeah. I think so. Yes. In short, I think they do go away at some point. Having said that, I don't think passwords are all bad, right? I think that there is a lot. Obviously, we get a lot of flack and for justifiable reasons. But there are some benefits to it, right? There is some value to passwords, which we can talk more about.

But how does it go away? What does that look like? I think like something that I'm personally excited about or that I think is a good replacement is biometric authentication. I think you haven't seen that ubiquitously yet, because it's a very new development. By biometric, I mean, like face ID, fingerprint, all of those kind of things. And that's getting more and more baked into the hardware products today, right? Like, your hardware devices. Maybe 10 years ago, that was not true at all. Like, five years ago. It's true now. It's true for high end devices. It's true for most devices. But it's getting to the point where Biometrics is baked into your device. And I think once that's ubiquitous enough, you'll start seeing a lot more apps simply maintaining or making that the default.

I think it's a great user experience. It takes like half a second to unlock your phone. Or like even my laptop has a fingerprint ID on it. And it's like very quick. You don't need to remember anything. It's very hard to spoof, right? It's like someone can't really – I mean, it depends on the underlying hardware technology. But in general, they're built to not be easy to spoof. That's something that I think could be exciting.

Having said that, that is in a way its own password, right? It depends on the definition of password, because that is still a password, right? That is still something that's uniquely identifiable that's being sent over the server to identify. But yeah, I mean, that's that good.

[00:35:37] SF: Yeah, yeah. I think my password, I mean, I would love to get it so that my mom is no longer writing her passwords down in a notepad that she keeps – because remember her password. But yeah, I agree. I think biometrics is something that has a tremendous amount of potential. And the fact that everyone – most people now, at least in the developed world, are carrying around essentially a computer in their pocket that can read their fingerprint, or do face ID in combination with things like secure enclave on the device. You can build a pretty secure system that's also really user friendly and a great user experience.

[00:36:10] AR: Yeah, for sure.

[00:36:11] SF: What's kind of next for SuperTokens? You mentioned you have a public roadmap. Are there certain items that you're working on that you would like to highlight and share?

[00:36:19] AR: Yeah. I mean, one thing we work on all the time is like the developer experience. And people talk about features all the time. But we think that to do developer experience correct is like got to always roadmap it. And there's always something to build there that's exciting, right? I think we spoke earlier about the CLI tool. That was like a project. We built the CLI tool, where there's like a lot of effort is now going into like there's docs restructuring and redoing the navigation and architecture of our docs. There's like always like some "feature" that's being worked on in terms of the developer experience.

Besides that, in the more typical traditional sense, we're working on multitenancy, which is like the ability to have like multiple user pools with one instance of SuperTokens. Or if you're a business and you have multiple customers, then you allow each of those customers to sign up to your app in their own individual way, right. We're allowing you to be able to do those things. We're adding the features to that.

We have two FA, but there's a particular form of two FA called TOTP, right? Like what you see with the Google Authenticator app, right? Adding support for that. Then there's adding features to our user management dashboard. There's this feature called account linking, which is a lot more complicated than it sounds. But it's like a user experience thing. We're working on all of those things.

[00:37:33] SF: Yeah, there's a lot to keep up on, I imagine. And I love that you're sort of carving out space to focus on developer experience. That's something that's very core to my interest in expertise as well. And like I mentioned, with community, I think it's not something – it's not a feature. It's part of the culture of the company to do that right. As we wrap up, is there anything else that you'd like to share?

[00:37:55] RP: Yeah, sure. There's this one interesting library that we built earlier. And this is just one of the things that surprised us while we were building SuperTokens. It's called browser tabs lock. Essentially, it's meant for locking. Sort of doing locking across tabs on your browser. And this is useful in session management when calling the refresh API. You want to make sure you call the refresh API just once in synchronous with all the tabs that the user might have opened to prevent some false positives when it comes to session hijacking. And as it turns out, this library was then used by all other auth providers as well, which was totally unexpected for us. And that's just a cool trivia about SuperTokens that we have this thing.

[00:38:41] SF: That's great. Yeah.

[00:38:42] AR: That library has like 600,000 weekly downloads now because they are used by everyone.

[00:38:46] SF: Wow! Well, congratulations. That feels like a really strong positive signal for what you guys are doing and the investments that you're making as a company. I want to thank you, Adi and Rishabh, for coming on the show. You definitely piqued my interest. Next time I'm building out a product where I need authentication or a side project even, I'm definitely going to check out SuperTokens. But best of luck with continuing to grow the company. I think it sounds really, really interesting, really exciting.

[00:39:11] AR: Yeah. Thank you so much, Sean. And you know where to find us, supertokens.com Discord. If you ever want to join the community, talk to us. We love talking to people. Feel free to join that.

[00:39:20] RP: Yeah.

[00:39:20] SF: Awesome. Thank you.

[00:39:21] AR: Awesome. Thank you so much, Sean. Bye.

[00:39:21] RP: Thank you so much. Bye.

[END]