EPISODE 1342

[INTRODUCTION]

[00:00:00] KP: Web applications often have some sort of login system. And once a user creates an account, they have access to features anonymous users can't see. In time, application designers will often add an admin level of access for special users. This is often the start of a slow trickle of technical debt. Proper execution of a programmatic authorization system requires concepts like roles, resources, departments and organizations. OSO describes itself as batteries included authorization. It's an open source library used by companies like Intercom and Wayfair, which allows them to manage authorization in a robust and standardized framework without reinventing the wheel. In this episode, we speak with Sam Scott, CTO at Oso.

[INTERVIEW]

[00:00:54] JM: Sam, welcome to the show.

[00:00:55] SSS: Thanks very much for having me.

[00:00:57] JM: Permissioning, authorization, there's a wide variety of tools that are in this space. And the first one that comes to mind is Auth0, which was more recently acquired by Octa. Tell me about a brief history of authorization as a service.

[00:01:18] SSS: Yeah, absolutely. So I think it's to start with, you need to disentangle authentication and authorization through very similar sounding names often lumped together as just auth. And there's often a very blury line between those, and which piece of the puzzle different people are doing. And even goes as far as headers, which the name authorization be used for authentication and things like that.

I think when you think about companies like Auth0 acquire by Octa, a lot of the stuff they focused on is primarily the identity piece, the authentication piece, right? So authentication is around identifying who the user is? Who you are? Checking some kind of credentials, that kind

of piece of thing. Authorization often the piece that comes afterwards. It's like now that I know who you are, what can you do? And a lot of the existing services out there like **[inaudible 00:02:12]**, they do a lot of the identity stuff and they maybe do a small piece of the authorization. They may be handle things like groups, or maybe it's like pulling a few attributes out of the latency providers who kind of get some sense of who this person is. But they often leave a lot of the authorization to the application code itself. It's like, "Alright. I know who you are and maybe I know what groups you belong to or role you have, but I'm going to let the app decide what to do with that information."

That's where we kind of see a lot of these companies kind of fitting into the picture is they handle that kind of front door, pop-upper, login dialogue, get you to authenticate or log in with Google or Facebook. And then they like produce some sort of like token, send that through to the app, and kind of like their job is done. They got you in the front door. They passed on some info in there. Kind of have to leave you do it.

[00:03:01] JM: How did you get interested in this space?

[00:03:03] SSS: I got interested in the space, more generally security. I did a PhD in cryptography. That was always driven by me just wanting to do like more and more applied versions of maths basically. Thought about doing some more like theoretical math, and they kind of want to get some more apply. And I was like, "Oh, this is cryptography thing. It's like practical math." Did a PhD in cryptography and was like, "Oh, I could try and do like more applied cryptographic research." And it just kept on leading me like down the identity of a herd this whole titles all the way down expression. But I kind of kept on like chasing that, like I want to be more applied all the way down that stack. And ultimately realized that a lot of security comes down to like do you know who you're talking to? And do you know what they're supposed to do?

Cryptography is often used as just like a mechanism to make that happen. But that's kind of how I ended up in this area. And more generally, when I'm starting out, starting out a company, the company Oso, for me the big thing was a lot of these problems like have technical solutions, especially in like in academia, a lot of this stuff is like solve problems. But the tooling around them is just like really hard to use that it doesn't feel sold for a developer. Like you have to go through a lot of hoops, you have to go through a lot of steps to make it happen. That's kind of

more generally what I kind of what I wanted to go out and try and solve. And ultimately why I started this company, Oso, just to make this whole area of security more accessible for developers.

[00:04:26] JM: Yeah. I felt like Auth0 getting acquired was kind of a small heartbreak for me, because the space was so much bigger, and they could have done so much more. I felt they sort of got distracted with the whole functions as a service effort. Do you see what they were doing there? I thought that was a very, very bad decision. And I hate to say that, because I think they were sponsoring us at the time when they launched that. And we promoted it.

I thought it was an interesting idea. Like, yes, you do want Lambda-like functionality around your authorization. Absolutely. There's nothing wrong with that. But there's probably a whole lot more that they could have done that's just deeper and more integrated into the authorization stack rather than trying to expand this functions as a service kind of thing.

[00:05:11] SSS: I think there's a really interesting sort of tradeoff they're probably fighting with, which is a lot of things in this space and security, you really do want to try and do it as this kind of like middleware like approach, which is achieve all with authentication. It's kind of like a front door. You log in, and then you have a token. And the app barely even needs to think about it anymore. And that's like so great from a security standpoint if you can just say like anything that got passed the gateway is authenticated. It's like we're good.

So it's like very tempting to want to do that with authorization and be like, "Well, what can we do at this level? If you're already in their checking identity, what more can we do?" But ultimately, you can only really end up doing somewhat coarse things, which the rules kind of functionality could give you, checking scopes in the token, checking API endpoints. But as soon as you want to get down to like what can this user do, like you're going to hit the database at that point. You're going to be basing on application data, and it breaks that clean separation.

[00:06:09] JM: So let's say you were in charge of Auth0 around the time when they made this foray into functions as a service around authentication. And you could choose an alternative strategy. What would you have done? How would you lead the company towards a direction that is not acquisition by Okta?

[00:06:31] SSS: What a question. I mean, I had to critique their approach. I mean, it does a fantastic excerpt. And I think they did incredibly well.

[00:06:40] JM: Could have been so much bigger.

[00:06:41] SS: It could have been bigger. The funny thing is I've spoken to other people who have been in the space that I probably can't name names, but who wanted to work on authorization, but they're like, "Well, authentication is so broken. Let's just go fix that one first."

[00:06:54] JM: Wait. Wait. Wait. Sorry. What the interesting authorization authentication?

[00:06:57] SS: Authentication, right? Checking who the user is. Authorization deciding what they can do. But like a lot of what Auth0 would do is that authentication piece. It's like integrating with single sign-on solutions and managing user accounts and tokens and all that kind of stuff. And they dip their toes into the authorization space with things like your rules and that piece.

Authorization, it's a pretty different problem domain. Like they're very similar. Like you check who someone is, and then you decide what they can do. The interface between those two is very tight, but they are like two very different domains. So it's hard for them for having a pretty great excerpt.

[00:07:35] JM: Hard to follow Auth0.

[00:07:37] SS: Yes. Hard to follow Auth0 for having that– For getting acquired and all that. I think I can see the value in what they do with the real stuff and being able to do some amount of like course authorization at the front door. I think that's like pretty reasonable. I think to do authorization, they would have had to do something completely different. And interestingly, they have – There has been a thing they've been working on through just this like Auth0 Labs thing, which is kind of more of a foray into the authorization space.

[00:08:03] JM: Okay, so give me some more details. Like, again, not to put you on the spot. But if you would have been Auth0 back then, you have traction, what would you have done?

[00:08:11] SS: I mean the problem is we're getting up against a kind of a difference in worldviews here. And we –

[00:08:18] JM: I love it. I love it. I love it. Take me on the detour. Take me on the detour. Your background, by the way, is New York on a rainy evening. It's the kind of place I want to take a detour in.

[00:08:30] SS: Yeah, let's go grab a hot dog.

[00:08:32] JM: What is that? A hot dog stand in the background?

[00:08:34] SS: That's a hot dog stand.

[00:08:35] JM: This is the best virtual background I've seen. Is this a Zoom native virtual background, or is it something else?

[00:08:42] SS: This is not a Zoom native. It's just a New York cinemagraph that I found just –

[00:08:48] JM: What? Searching for cinemagraphs. Okay.

[00:07:58] SS: Cinemagraphs. Yeah. Okay. For those who know me, I've been using this like blowing in the breeze grassy background for like the last like 18 months of the pandemic trying to give myself some Zen. I am based in New York. We briefly went back into the office for a bit and then COVID forced us to shut back down again. So I have my New York background to make me feel like I'm still here.

So the detour we want to take. About two years ago, Google released this paper on this authorization system they built internally called Zanzibar. It was this – I think they called it global scale consistent authorization system, something like that. And it's effectively something they've been working on for. Yeah, I think it started pretty early 2010s on how to solve authorization at Google scale, which they shared some details in the paper. It means handling something like 10 million planned requests per second, right? 10 million authorization decisions per second.

So Google had this problem of like how do we solve this in a way that's scalable for our needs and also consistent to use for developers and people building apps like Google, right? So you're building Google Docs. You're building Google Drive. You're building YouTube. Like all these different systems need to do authorization. Like how do we do this?

So they released a paper in 2019 that kind of detailed that. And it's sort of made up of two main parts. There's like one is this like authorization model, like a configuration language for how to express authorization. And the second is like the system itself built on top of Spanner to be able to handle the incredible amounts of volume. And I think what happened after they released that paper is a lot of people were like, “Huh! That's interesting. Like maybe there's something here. Maybe this is like an approach we should be taking to building authorization. And so we've seen kind of in the last half year in particular, there're been various tech companies who have tried to emulate this. Airbnb wrote a blog post on this recently about how they have their own version of this. And you have companies like Auth0 and other startups who are trying to build this Google system to kind of expose it to the world. So coming to the worldviews piece of this.

Now, the fundamental thing that kind of Zanzibar asks you to do is to put all of your authorization relevant data in a central service, in a central place, so that it can do authorization over it. And that means everything like what organization does a user belong to. What organization does this file belong to? What folder is this file inside? Who created this video? Who's been invited to edit this document? All of that information that you're going to need to use decide who can do what goes inside this like central service. And then from there, the service can now start making authorization decisions, because it has all that data.

I mean, the problem we see with that is – And requires you doing two things. Like, fundamentally, re-architecting your app so that it can put all this data into a central source. And also like trusting that external thing with all your data. And I think this is really appealing, especially from a like startup business standpoint, because services are great. People kind of understand like what they're paying for when they get a service. They’re just like I'm going to pay you to manage all these things for me. It's like easier to build a manage when you're a company, because you can push updates to the service and so on and so on. But be it Oso about the right way to solve authorization is you shouldn't be re-architecting your entire app

around authorization. You should be building authorization around your app. And that means using the data where it lives inside the app and being able to make authorization decisions based on your existing data models, not by re-architecting everything. That's sort of the big – I'd say the big clash in worldviews. And I think, in particular, it makes it hard for me to answer this question of like what would I do if I was Auth0, because they are set up to be a SaaS company and to manage the stuff on user's behalf. So for them to kind of take this alternative approach where the app data stays where it is would be a pretty big change in direction for them.

I don't think if I was the CTO at Auth0, I'd like recommend such a dramatic change in strategy. But as someone who started on this problem two, three years ago from scratch, and thinking about what's the best way to solve this for developers – So we're doing this from first principles in some sense. That's kind of what the decisions we landed on.

[00:13:24] JM: So I did Sequoia Scouts for a while. So I'm familiar with how selective Sequoia is in their investing. You are a Sequoia portfolio company. That is a rarity. How do you get this product to a point or this vision to a point where Sequoia wants to put money into it?

[00:13:41] SS: So growth investors, they take a long term view on things, right? And that was kind of one of the reasons we were kind of very excited to work with them. And I think they get the size of the problem space that we're going after, right? Every like commercial enterprise application, the vast majority of like direct to consumer apps are going to do authorization. And this is an area that people have mostly just built themselves from scratch every time. So when we talk about like the scale of the problem here, authorization, this is as big as like you don't build your own database. Like you don't build your own payment system, right. That's the size of the problem we're talking about. And I think they see that vision and they see that opportunity.

[00:14:20] JM: Well, that's very concise. Can you go into a little more detail?

[00:14:22] SS: I mean, which piece?

[00:14:24] JM: I mean, come on, man. You're an open source. How would you put your company in a nutshell? Open source authorization as a service, right?

[00:14:32] SS: Currently, we're a library. We're currently open source library for authorization. We're sort of more of a framework. We call ourselves it's a batteries included framework for authorization. There're kind of a few pieces in that. The framework piece is like we are going to give you a structured way to think about authorization. And there's like a lot inside this problem that people don't even kind of realize it's pieces. It's like what's that like logical piece of it? Like how do you end up making a decision? There's the bit that you add to your app. There's the way that you integrate your data and all those pieces. So the framework is kind of like this foundation for just even thinking about this problem in the first place.

The second piece, the batteries included, is we're going to take like everything we know about authorization, all of the best practices, how you should end up thinking about exposing this to users, structuring it, modeling it, all this stuff. I'm we're going to bake that into the product.

So when you come, when you sit down, you're building an app and you're like, "Well, how am I going to decide if this person should have access to this resource?" You're not reinventing the wheel. You're not doing this from scratch without any guidelines. You're getting like the product telling you like, "Hey, you should really think about roles. Roles are very useful. And here's how you can do roles in an organization. And if resource wants an organization, like here's how that structured. Here are the data schemas. And look, that's the ones you have in your app. And here's how you think about this. And really building – Putting that all into the product so that it goes from being this unstructured problem, which can be very complex and hard to think about. And if you get it wrong, you have to rewrite it and people end up refactoring their app four or five times to get it right. It goes from that to being something that you can just like sit down, get right. It doesn't even seem that hot anymore. You just sort of express naturally what you want to happen. And that's what the product does.

[00:16:14] JM: So a perfect example of a customer is Intercom. When I think about the kinds of authorization problems that Intercom has, it's quite complex, because you have a user that lands on a web page, and they need to engage with the Intercom chat system. That's all you need to know to know that this is a very fine-grained permissioning problem, at least in the sense of intercom, right?

[00:16:38] SS: Yeah. The real context of Intercom is the piece of that you don't see on the website, but who's allowed to get access to that data of the customer talking with – Potentially talking to a live agent. Who can get access to those conversations? Who can see that? If this person's like maybe sharing confidential information about their accounts, trying to get some support. Like who gets access to that? Who can go in and see it and maybe assign it to other people, or ultimately end up handling those customer requests? That's where it can really start getting complex, is like what are the different ways that someone might be given permission to view a particular piece of data or a particular conversation?

[00:17:16] JM: Okay. And so how do you build something that Intercom is willing to adopt that is extraordinarily tightly coupled with their infrastructure? Because Intercom is a mature company. They're sophisticated engineers. They're not going to adopt something like this overnight.

[00:17:31] SS: Yeah. I mean, I think the biggest piece of this, and this kind of comes back to where the data lives. The thing that was easiest for them is they didn't need to do any migration work in terms of the data. There's no need to like, "Okay, first, we're going to go and upload all the data of service from React to our app to make this happen." They could get started with that immediately, because they import the library, they add it to their app, and they have access to everything that they have access to in their app already. And that was like the easiest thing. For them, that's just the simplest thing to get started. They leave the data where it is and they can immediately just start using Oso for like just trying like building stuff on top of it.

So for a larger company, I think the big part of this is taking something that is not very consistent across the app. It's kind of maybe done in different ways in different places. Kind of consolidating that down and having a single place to understand how authorization is modeled, and putting that in like a single policy and be like, "Okay, I can now see like the structure of this in our entire app, instead of chase down the code path to find out." So that's kind of like the first piece.

And then from there, it's a case of like seeing the new capabilities they can add, the stuff that they couldn't previously do and how easy it'd be to add this. They want to go more fine-grained and start adding things like attribute-based access control. Can this user see this resource

based on attributes of the user on the resource? It's like incredibly simple for them to add that, and they can kind of go and POC it and see that it works. So it's kind of like a – At the end of the day, it was like a relatively small amount of work for them to migrate over. I mean, it was largely just done by one engineer who went through and tested it. And they actually ended up being able to delete a ton of code by having this by kind of consolidating it down. And then there's like a bunch of new functionality they can add without having to undergo and go through like a lengthy design process.

And I think that's like a really good example of what like any company at any stage will go through multiple times throughout their life of authorization. Like to start with, I just need like admin member. But then they start adding roles, and they start like having to add projects, and teams, and groups, and all these things. And each time it often ends up being like a re-architecture. Whereas like Oso is flexible enough that you can – Really, it's just a case of integrating it with your existing app data, and then expressing the logic you need.

[00:19:43] JM: Okay, so is this something that I would use in my own application? Let's say my random application. Let's say my random web app. Let's say I'm building softwaredaily.com, which is one of my favorite websites, if not my favorite website. It's very basic. Actually, we have a pretty terrible login system that currently doesn't even work. And I believe somebody is injecting some kind of malware code into it. I hope that's not happening. But if you go to softwaredaily.com, you're going to be logged out. We have a kind of crappy login system. Yeah, actually, maybe we should use Oso. Can you tell me if we need Oso? Can you go to softwaredaily.com right now and just look at our login system and tell me if it's like terrible?

[00:20:25] SS: I mean, probably **[inaudible 00:20:26]** from the login piece itself, say we handle the authorization piece for everything that happens after your authenticated. Login a single sign on. Sign in with Twitter. I mean, that's usually a pretty good start.

[00:20:37] JM: Does that work for you? See if that works for you. I think it's broken. Do you have a Twitter account? Do you use Twitter?

[00:20:41] SS: I do. But it's asking me to authorize a lot.

[00:20:44] JM: Yeah, that's a problem. That's a problem. We definitely need to fix that. All right, request for open source contributors. **[inaudible 00:20:50]**. All right. You definitely don't want an open source project to be able to delete tweets from you.

[00:20:57] SS: Yeah. I know. That doesn't sound good.

[00:20:59] JM: All right. So forget it. Do you have a burner Twitter? You use a burner Twitter or log in with burner? It doesn't even make you go check your inbox or whatever. So you can put in a fake email. That's fine.

[00:21:07] SS: Let's try that.

[00:21:08] JM: All right. Nice. All right, cool. You can post a company or topic, you can post a job, you can do some writing. There're some various things. Eventually we'll need admins, right? So I think once we need admins, then we probably need the authorization, right? Admin is an authorization issue.

[00:21:22] SS: Yeah. Alright, check this out. Okay, so I can see a couple of things. So pretty nice example is maybe I can like edit or delete posts if I wrote there. Or if I'm posting jobs, I probably want to be up to control if I want to go and edit those things.

[00:21:38] JM: Click on one of these topics. Click on community and products thinking at the top. For example, if you click on the little pencil on the question at the bottom, the question, "How can we make it less unclear that will happen right now?"

[00:21:50] SS: I didn't have a pencil for this, though.

[00:21:52] JM: Oh, really? Oh, okay. All right. That's right.

[00:21:54] SS: Good work.

[00:21:55] JM: That's right. Okay. But that's exactly what you're solving, I guess, is who has the pencil?

[00:21:59] SS: [inaudible 00:21:59] become a maintainer. What does that do?

[00:22:03] JM: You should be able to do that.

[00:22:05] SS: Yeah, I have become a maintainer. I'm sorry.

[00:22:08] JM: Right. That's great. That's great, man. Congratulations. Change the topic. I think your product is exactly what we need for this, because we need fine-grained permissions. Basically, if you think about Wikipedia, right? Wikipedia has this problem. Wikipedia needs fine-grained permissions in their open source thing, right? Like Wikipedia is a perfect example. You need a hierarchy of authorization in Wikipedia.

[00:22:42] SS: Yeah. So you'll probably just see the fantastic test-test account just updated that, say we need to update our authorization.

[00:22:51] JM: On no. Alright. Hold on. I'm going to add a highlight to that. I'm going to say no, we don't. I need to log in real quick. Let's break authorization right now. Wait, okay. Sorry. Hold on. Sign up. By the way, if anybody wants to know what we're doing, you can just go to softwaredaily.com. Wait, what is this Twitter thing? Return to Software Daily. What the hell is this? Why is there a Twitter interlocutor here? This doesn't make any sense.

[00:23:14] SS: Okay, so like there's **[inaudible 00:23:15]**, right? So if you want to sign in on Twitter, you don't know – You would like know who the person is logging in as from Twitter. So you're basically asking, "Hey, Twitter, can I read this person's like account information so I can identify them?"

So Software Daily needs to ask Twitter, "Am I allowed to get the login info for this person so I can like see who they are?" So when you set up the developer like integration on Twitter, you need to choose like what you want to get access to so they can prompt the user for that consent. But it sounds like you asked for a lot of capabilities, like posting on someone's behalf. **[inaudible 00:23:50]** click on a topic and maybe Software Daily could post to Twitter on my

behalf and things like that. That's why you're getting that big, like scary consent page. Because you basically asked Twitter, “Hey, can I do all these things on the part of the user?”

[00:24:05] JM: I left a highlight on your alteration, by the way, if you want to refresh. You can respond. I'm going to up-vote it. Honest question. Do you think this website is useful?

[00:24:15] SS: I mean, as a very biased person, I’m not sure whether I should post our jobs here.

[00:24:21] JM: You should do that. Post your product. Post your product right now, Oso. By the way, anybody listening, go to softwaredaily.com, post your product or whatever, if you can log in successfully.

[00:24:31] SS: I’m a little worried that someone will come in and start editing it and trolling my posting now that I see that it can add anything.

[00:24:36] JM: That would be great, because that would force us to adopt Oso, right?

[00:24:40] SS: I'm not sure that's the right ordering. Normally it’s big impressive company, in that case, us, who wants to use your product and you're like, “Oh, we better solve the security things before they use us.” That's usually how the security thing goes. They're like, “Well, we really want to get these enterprise deals. So we better like fix our security so that they'll use us.”

[00:24:59] JM: There is one person that uses this website, and that's me. Actually, there's also – If you go to the website, there's something at the top called Talk to Strangers America. Do you see that? That's another user that uses our website. I don't know what this is.

[00:25:14] SS: It came by random stranger chat. I love that. Maybe I should become a maintainer of this as well.

[00:25:21] JM: You should. Oh my God! Please. You can add some jobs this page too, if you want.

[00:25:27] SS: Yeah, let's delete all the other jobs as well. So the answer is up.

[00:25:30] JM: There's no other jobs. I don't think Talk to Strangers in America is hiring anybody. You mean jobs in general? Yeah, there are –

[00:25:37] SS: Just one in general. Yeah.

[00:25:39] JM: Yeah. Anyway. Okay. Well, let's get back to Oso. No more self-promotional, like thinly veiled self-promotional Software Daily promotion. Oso, yeah. So batteries included authorization. So you've got like an open source authorization framework. This is pretty cool, man. I got to say, congratulations. This is pretty cool.

[00:25:57] SS: I don't know. Authorization is hardly the most sexy topic in the world.

[00:26:02] JM: What are you talking about? Come on man. IM policies – IM policies are a huge headache, aren't they?

[00:26:07] SS: They are a huge headache, and nobody likes them. I mean, no one wants –

[00:26:10] JM: Nobody likes them.

[00:26:12] SS: You just want it to work. You just want to like deploy your app without getting weird IM errors.

[00:26:15] JM: Yeah. It’s ghastly.

[00:26:18] SS: Yeah. And I think the piece where it becomes fun, right? And we don't want to take away like all of the like craft and like sharp edges and all that stuff where it gets like annoying for a user. Make it so you're building softwaredaily.com and you're just thinking through this process of like who can become a maintainer? And you just want to say – It's just like no one. No one can become a maintainer. Or you can become a maintainer if your email address ends in @software.com. It's like, cool. Now you've done that. You've got this like nice bit of logic that only people with the right email address can be maintainers. And they can make

themselves maintains with things. You can start saying – You can edit a post if a post belongs to a topic that you're a maintainer of. And like that piece of logic there, right? Which is it's using bits of the app, the topic and a post that belongs to a topic. You've probably got some like one to many database relationship where it's like stored in whatever, in your database, and you just to like navigate those relationships to see the posts. That is like two lines of code with Oso. It's like you just say, “Yeah, you can see this thing if you're a maintainer of the post that the –” You can see the thing if you’re a maintainer of the topic or the post belongs to you. It’s like two lines of codes. It’s like that easy to kind of represent that.

And that does get fun, where you're just like thinking about kind of like, “Oh, how do I model this? Like how do I flow things through? Does it make sense for me to have like post roles? Like should someone be assigned as a reviewer of a post? Or should that just like flow from something else? And how do I want my users to interact with this and move through the website? Should I hide the – If I need to hide the pencil because someone's not able to edit the thing, then how does my frontend know that the backend is not going to let me edit this thing?” Doesn't look as fun. That's like just product design, product feature design, like thinking about how your app is going to work. And so we want to make that the piece that you focus on and like all the other stuff around — like trying to navigate the security world of acronyms. It's like RBAC, and ABAC, and PBAC, and MBAC and being like –

[00:28:15] JM: Yeah. **[inaudible 00:28:16]**

[00:28:17] SS: No one wants to deal with that stuff and no one wants to like read newspapers on hierarchal roles. They just want to like think in like what’s mapping to do? That bit is fun. That bit I do agree is fun.

[00:28:28] JM: You're like a child. How old are you? 28? 25?

[00:28:32] SS: I'm 32.

[00:28:34] JM: You’re 32. Okay, you're like my age. Sorry. Oh, I see. Okay, you got a pure PhD in 2017. Okay, you graduated like the same year I did. Nevermind. Actually a little bit –

Well, yeah. Same year I did. But I was a little bit slow. I spent like five years in undergrad, or six years maybe in undergrad.

[00:28:49] SS: I did some traveling in between. That was my –

[00:28:52] JM: Nice. So you got a PhD in cryptography. Let me ask you like naïve, like I'm a reporter for like some crappy magazine. Question, if you're so good at cryptography, why aren't you building cryptocurrencies?

[00:29:04] SS: It's funny, because yeah, I mean, I was doing my PhD like just as as Bitcoin and cryptocurrency was taking off. Like we went through my reading groups, all the blockchain papers and stuff. And obviously, I was like there's probably a forest for the trees thing going on where I like understood the technology. I was like, “Oh, that's really cool technology. But I don't understand it.” No, you could use this thing. This isn't a real thing, right? Yeah, I mean, I think – I'm pretty sure like a laptop that I throw away many, many years ago here's some Bitcoins on it from back in the day where you could just click a button and get a Bitcoin. Let's not dwell on that too much.

Why am I not working on them now? I think there are some interesting problems. It's just not the path they went down in terms of the kinds of problems I care about solving. I think there're some really amazing ideas especially around –

[00:29:51] JM: It's a fake technology. It's not a real technology. It's a data structure. The blockchain is an interesting data structure, and everything around it is a fake technology. You know what's funny is everybody was saying it's blockchain, not Bitcoin. And then all the Bitcoin people were saying, “You're wrong. You can't have a blockchain without Bitcoin.” And they're actually the wrong ones. The blockchain is the most interesting part of cryptocurrency.

[00:30:15] SS: I mean, I don't know, decentralized consensus-based software is I think is a super interesting –

[00:30:20] JM: But here's the thing. You can decentralize across cloud providers. Nobody's doing this. You don't need to decentralize across a bunch of random data centers in China, and Uzbekistan, and whatever. You don't need that. Just decentralize across cloud providers.

[00:30:37] SS: Yeah, I agree. One of the things I did work on during my PhD is this idea of like password hardening through like a third-party service. And the general idea was like it was using these like kind of cool cryptographic measures that meant that there's a key on the server, which is like the server that has the keys doesn't know the passwords you're sending them, because those are like properly blinded and hidden. And the server that manages the passwords doesn't know about the key. So those two things like entirely separate. And then you can even start doing like cool things like you'd have like multiple keys and distribute them across multiple servers, and things like that. And that does mean that you can like protect your data. So like your password database, your user's passwords. You can protect it by keys that are managed on a bunch of third-party servers across multiple cloud services.

And if any one of them gets compromised, you can rotate the key and keep going, and like that key is worthless. It's got really, really cool, strong security properties. And that was the kind of thing when I was on my PhD, I was like, "Wouldn't this be a cool thing to go and like build and help the world secure their passwords and things?"

And then I went and spoke to some companies, and they were like, "We should be encrypting our passwords? Or what's this hashing thing I've heard of?" I'm like, "Oh, you're not worried about like they act as compromising your password databases. You just want to make sure that like someone just downloads your entire Postgres database, then the password is not like completely revealed." It was just like very different like threat models.

And so there are some cool cryptography you can do, whether it's, yeah, key management split across multiple services, or it's like some, I don't know, blockchain-based decentralized algorithm thing. I mean, 99% of companies, that's not relevant to them. I mean, this is relevant for like governments and militaries and defense contractors and stuff like that, not, no offense, softwaredaily.com.

So, I don't know, I just saw there's kind of an opportunity to make security better for the vast, vast, vast majority of people where it's just simple things like protecting access to data on the like web servers and not worrying about like state actors and all this like fancy cryptographic stuff, but it's just like just kind of making the stuff that is already solved like easier to use.

[00:32:59] JM: But seriously, though, think about the blockchain as a data structure across cloud providers, DigitalOcean, AWS, Google Cloud, Azure, name 50 other cloud providers across the world. There're ton of cloud providers that nobody's heard of, Equinix, whatever. You've got a blockchain across all of them. You just synchronize the data. You've got a great blockchain. You've got great shared infrastructure. Because here's the thing about blockchains, it's a shared finely-grained permission database. That's what a blockchain is. You don't need to use it for currency. You can, but you don't need to.

[00:33:38] SS: Sure. Yep.

[00:33:41] JM: Isn't that an interesting application platform? You can do anything with it. Everybody's excited about smart contracts. What? Just make programs on top of a blockchain.

[00:33:54] SS: I mean, there are a lot of people who are doing one of like DAPs, decentralized apps. Like that is –

[00:33:57] JM: Yeah. It's like, "Dude, I'm doing a decentralized app. It's called Software Daily." It's on three different Heroku servers, I think, right? Or at least one. It has some sort of redundancy. Like what is this fraudulent industry?

[00:34:12] SS: That's an entirely separate thing, right? The merits of the technology is kind of separate. I don't know if I'm allowed to say this, but it's as a bastardized version of the technology, right?

[00:34:25] JM: It's distributed systems **[inaudible 00:34:27]**, or distributed systems – I don't even know what to say about it. It's like take Leslie Lamport and introduce his worst nightmare.

[00:34:36] SS: It's been co-opted by more than just like the technology now. The speculation on cryptocurrency isn't so much around the technology anymore. I think there is a lot of people who do deeply care about technology and the things that can do, but that's not why it's become this like speculative investment pool with no safeguards. There's kind of like two trends going on here. There are people who genuinely care about technology and interested in building interesting stuff that has good privacy guarantees, for example. There are banks who are like building consensus things as like a bank to bank thing. And it's in that kind of model you just described, right? Where instead of cloud service providers, it's just like banks doing consensus in a more modern way as they're just like sending each other ledger on paper or whatever. There are people who do care about technology. And then there is like cryptocurrency, which is just become this vehicle for people. I don't know. It's just an entirely unregulated, speculative investment thing that anybody can partake in and very dangerous and damaging to the environment.

[00:35:42] JM: Hold on. The damaging to the environment thing is almost as dumb as the industry itself. The argument that this is damaging to the environment it's the dumbest thing I've ever heard. Why is that a good argument? Like are data centers bad for the environment?

[00:35:56] SS: Well, I mean, it’s like yeah. I mean, but it's a utility question. What are they being used for?

[00:36:02] JM: All kinds of things.

[00:36:04] SS: I mean, do you think cryptocurrencies have been used for all kinds of things? Or are they being used for –

[00:36:09] JM: Well, I mean, cryptocurrencies are hilarious because they're mostly being used to defraud cryptocurrency investors.

[00:36:14] SS: Right. I guess is your point that like why care about the environmental aspect if it's already a terrible damaging thing?

[00:36:22] JM: It's like lots of things are damaging to the environment, right? Like Coca-Cola. Doesn't Coca-Cola like pour toxic sludge into rivers? Aren't there like places in Africa where just like big corporations like Nestle – Doesn't Nestle pump toxic sludge into like random African communities and cause birth defects everywhere? I mean, is this anything new?

[00:36:46] SS: I think the thing is – Right. If you're going to split blockchain cryptocurrencies into those two things, like the legit technical usage and the like cryptocurrency as a way of defrauding investors, if you're going to split it into those two things, then even the technical stuff becomes a little bit wilier on whether it's like good because of the environmental impacts. It's like the scale that – The number of like legitimate transactions that Bitcoin can handle to process like online transactions, it uses like ridiculous amount of electricity compared to like the equivalent that like the visa network can support. And so I think it's a reasonable point to say why maybe the technology is not as great as people think it is. I think it's relevant in that domain. But if you're talking about just like purely like is cryptocurrency a good thing or not? And ignoring the technology piece and just the speculative market piece of it, then it probably doesn't factor in as much, right?

[00:37:44] JM: What's the hardest technical problem you're working on right now?

[00:37:48] SS: We have to like very hard problems that we solve at this company. One is very much product-focused, like how do we make like authorization concept simpler for developers? And that's purely like UX design, that kind of stuff. That's one was hard. But in terms of like the technical challenge, so we haven't really got much into it. But the way that the Oso product is built, it's available as a library for like multiple different languages like Python, Ruby, Java, Node, a bunch of languages. And it does that by having this like embedded policy language built in Rust. Alright? So when you express authorization in the policy language, there's this like common implementation, the kind of nuts and bolts of it under the hood. It's all being done by its common component. That piece is really, really fun to build.

In particular, though, we have this concept, this feature called data filtering. And the idea is instead of just saying, "Can this user access this thing?" You're able to say, "What are all the posts this user can view?" And the answer to that might be posts where the user is a maintainer of the topic it belongs to, and posts where the user was around who wrote the post. All these

different things that potentially can be used aside who can see a thing. So we're basically changing the language from here to just say yes-no things to be able to return like here are the things that need to be true, or our users have access. And returning those is what we call like constraints, right? And there's like a data representation for that that says posts where kind of like SQL, right? Like posts where post. created by equals users or ID, things like that.

We supported this for a while in Python with like a direct RM integration with Python. We're currently making that functionality like available that anybody who's it from any app by kind of like implementing this pretty simple interface. And that piece of technology is kind of mind blowing that you can write your authorization policy to say who can do what in your app? It can both answer like can I edit this post? Yes or no? It can also say what are all the posts I can read and return this like – And do it in a way where it's integrated with your existing application data. So these like constraints, they will get applied like inside your database as a result of like fetching the data. And it just sort of it makes this like beautiful boundary between authorization logic and the app data where your policy really does just represent the logic who can do on my app. Everything to do with the application day, like how do I fetch this piece of data? Like how posts and topics, like how are they related, stays in your app. Doesn't require you like refactoring anything. The functionality that provides is kind of mind blowing. And kind of bringing things full circle, that's like one of the features that like Zanzibar provides that I think no one really figured out how you can make that work without removing all your data into different service. We put that work in, and it's kind of mind blowing.

[00:40:40] JM: Okay. Well, we got five minutes, you're a pretty interesting guy. We should definitely do another show if you're down. If you're ever in the Bay Area, we'd love to hang out. Can I just ask, why aren't companies default open source? It's such a mistake, right? This company should just be default open source, right? Or if you don't have a plan around open source, you're probably making a mistake.

[00:41:00] SS: I actually don't agree. Personally, I love open source and believe in open source. And like all that goodness, I think especially for like security cryptography, like if we could have many eyes on a piece of security software and make sure it's harder to make sure it's been like analyzed and scrutinized by the community and great to have projects that people

can contribute to and so on and so on. I really strongly believe in open source from a personal standpoint and for like moral good of it.

I wouldn't say also dogmatic about it, that as a company, I was like, we must do open source because the right thing to do full stop. We did open source, because two things. One, we decided that building authorization as a library was the right way to solve this problem and it was the right thing to do for developers. And we put developers first. And then number two, if you're asking people to integrate a library into their app to do something as critical as authorization, that should be open source. People have the right to be able to see what's happening there and contribute to it and potentially fork it and get to do the things that needs to. But if we had decided that this would be better as a service, I didn't know that doing open source would have been the best strategy, because it's hard to do open source. You have to maintain these projects. You have to – I mean, if you don't do it well, right? You need to welcome a community and support them and help them if they want to make contributions in how they do it. You're committing potentially to having multiple versions of a piece of software out there that you maybe maintain max compatibility with. It's hard. And so if it's not your like priority, because you're building something else as a product and you just open source as like a sideshow or something, you're not going to do it well. Like you can't split your focus that wide.

But flipside of it is doing open source is so much fun, because I love our community, and I love interacting with it. We have a Slack community where you can join. Just come and talk to us about authorization or the product more generally. And we did a hackathon, and we had community members join in. The upshots of it are amazing. They're a lot of fun, but it's definitely work. We spend a lot of time in our community and welcome people and making it open. I'm not sure I'd say you should like default to it, because you need to invest time in it to make it work.

[00:43:12] JM: Look, I would like to keep talking longer, but I really got to go. I have emails to attend to. Let me know when you want to come back on the show or whatever. And if you're in the Bay Area, let's hang out.

[00:43:21] SS: Great.

[END]