EPISODE 1500

[INTRODUCTION]

[00:00:00] ANNOUNCER: This episode is hosted by Jocelyn Byrne Houle. Jocelyn is a product manager, founder and investor. Today, she is an operating partner at Capital One Ventures where she focuses on data, ML, FinTech and enterprise applications. In the past, she has held product and technology leadership roles for multiple startups and for big companies like Fannie Mae and Microsoft. Follow Jocelyn on LinkedIn or on Twitter under the handle @jocelynbyrne.

[00:00:31] JBH: Enabling authorization policies across disparate cloud native environments is complex, and can be a roadblock for software engineering teams. Open Policy Agent, or OPA is a declarative policy as code approach to authorization that reduces security and compliance problems for engineering teams. The way it works is by translating business context into declared policy statements. These policy statements are then compiled into code and deployed as agents that can be injected into any process such as an API gateway, Kubernetes Provisioning Service, or Continuous Delivery Automation Service.

Styra created OPA and contributed it to the Cloud Native Computing Foundation, where it is a graduated project with over 130 million downloads to date. It's used at large companies such as Netflix, Atlassian, and Pinterest. The Styra Declarative Authorization Service, DAS, is specifically designed for OPA and it includes the enterprise great ability to author policies, preview impacts of new policies, and document all the history of old policies, all managed in a single pane of glass.

Tim Hinrichs, CTO and Founder of Styra joins the show today to discuss how to make authorization policies easier to author, distribute and monitor. He will also share with us some of his thoughts about leading an open source software company.

One note of disclosure to be aware of, Styra is a portfolio company of Capital One Ventures. The strategic investment arm of Capital One. Views and questions expressed in this podcast and related material are my own, or those of my guests, and do not reflect the views of Capital One Ventures or its affiliates.

[INTERVIEW]

[00:02:19] JBH: Thanks for coming to the show, Tim. I wanted to get started with some terminology questions if we can start with that. Maybe you can speak a little bit about the difference between authorization and authentication, and then bonus follow up question. Why are the name so similarly?

[00:02:35] TH: I wish I knew the answer to the second one, I don't know. Whoever –

[00:02:37] JBH: It seems like terrible marketing.

[00:02:41] TH: It's always confusing. Even people who like implemented this stuff, they sometimes flip and flop because the words are so similar.

Yeah, so authentication, I always describe this as how do I prove to a computer system that I am who I say I am. And usually that solved with like a login and a password, maybe a thumbprint, face scan, MFA if you're using your phone. So really, at the end of the day authentication is about who am I? Authorization is sort of the next phase of controls that you have around, let's say how people interact with software, which is, I go in, I've already authenticated, let's say it's my banking application. I go in to that banking application and I'll click a button, view a report, try to withdraw money. Every one of those actions that I'm taking needs to be authorized. The software at the end of the day is going to check like, can Tim withdraw money from his wife's account or not? Right? That's an authorization problem.

So, the way I like to think about it at a systems level is, authentication, that might happen, sign in. That might happen once a day, once every 30 days. I think I sign into my Google accounts every 30 days. Authorization is happening on every button click, every page view, every list of results that I get, all of those results have to be authorized. So, wildly different in terms of the implementation, and how hard it is to implement in terms of how frequently the authorization problems get solved, compared to the authentication problem.

[00:04:10] JBH: That's really helpful. I like that description, right about every kind of touch that happens inside that application. How does policy fit into that world of authorization?

[00:04:19] TH: Yep. So, policy is an interesting word. Sometimes I'll call it policy. Sometimes we'll call it rules, regulations, best practices, all those things fall. They all sort of mean similar things in the sense that, let's go back to the banking application example. That piece of software, that banking app, has to have some logic inside of it that makes a decision, like, can Tim withdraw money from this account? Can he see the list of report for tax season? That logic is the policy that dictates exactly which actions I can take within that piece of software. So, that's the policy. We'll call those rules those regulations. That logic, that policy, I'll use those all pretty much as synonyms, actually makes the decisions. Right now, there's software that's involved, that banking software has to like, ask, "Hey, does Tim have in fact, the permissions? Does the policy say he's authorized to push that button or view that report?" But the decision itself, we'll often call that policy, the logic that encodes that decision will often call the policy.

[00:05:15] JBH: I really love that description, because I think sometimes the policy gives it such a mysterious software feeling, when I'm talking to leaders about it. And really, it is this sort of just short set of instructions, right? For you all, you have a couple of big use cases that you focus on. So, help us understand how authorization plays a special role for a particularly, I would say Kubernetes environment and the microservices environment. Those are two use cases. I'd love you to talk a little bit about the special problems of authorization in those worlds.

[00:05:48] TH: That's a great segue in the sense that the way I sort of set up authorization in that description was you've got a banking application. As I log in, and I poke around at that application, there are all kinds of authorization decisions that need to be made. I'll often refer to that as the application level of authorization that could correspond to microservices. By the way, authorization could take place within the application, even if it's a monolith. There's still one authorization problem needs to be solved. And more generally, then I like to think about it as application level authorization. So, we've already described that.

The second one as you describe it, like Kubernetes, I'll generalize that a little bit and talk about it as the infrastructure level of authorization. And here, the idea is like, "Look, at the end of the day, authorization is fundamentally about people or machines, taking action on software." And

the software has a side. Is this action authorized or is it not? Is it allowed? Is it permitted or is it not? That same idea applies not just to the banking application world, or any application world, but also to the infrastructure. Because at the end of the day, developers who are building and running those applications are also using software. They're using things like Kubernetes and AWS, and CI/CD pipelines, and they're interacting with software all the time.

So, at the infrastructure layer, the authorization problems that you need to solve are really focused on developers, taking action. Maybe they're upgrading their software. They're spinning up a new server. They're connecting a new network. They're attaching a new storage volume. And the question that needs to be solved then, buying that Kubernetes cluster, buy that public cloud, buy that CI/CD pipeline is, is this developer authorized to spin up that new server, that new network, or that new storage system? Or are they not? It's the same conceptual problem, authorization, it's just applied to a different set of software, into a different, end user at the end of the day.

[00:07:33] JBH: Interesting. So, there's really – you're triggering me to think a little bit about the personas and the jobs to be done here, right? Which is developers provisioning their environments, IT leaders making sure that they're coordinating across multiple environments, and then at the application level, which I'm most familiar with, making it easier for those applications to run correctly, authorize the right people, that type of thing. So, users are getting an enhanced experience as well, that's more accurate. I think the interesting thing is the complexity of working – there are so many policies out there, and they're almost immediately out of date. Tell me a little bit about how OPA was intended to tackle this level of complexity. I'd love for you to talk a little bit about making the lives of developers easy, but also the kind of performance boosts you get by running close to the infrastructure.

[00:08:21] TH: Yeah, all good questions. We'll see if I can get them all. When we started Styra, the problem that we saw was kind of in part, I gave you like a two-level kind of an introduction to authorization policy, which is applications and infrastructure. It's true. But even within any one of those, you've got a multitude of different pieces of software. At the infrastructure level, we rattled off Kub and public cloud and CI/CD pipelines. But in reality, there are how many products at each of those levels? I mean, maybe there's just one Kub. But there are multiple public clouds. There are a large number of CI/CD pipelines.

At the application level, you've got all kinds of databases, all kinds of service meshes, all kinds of gateways. And when we started the company, what we saw was that all of those different products and projects have a completely unique way of solving authorization, right? As a user, if I've got 50 different pieces of software, I need to learn 50 different ways of solving, of like putting in place my own authorization policies, even though the policy, the logic itself may be the same for all of them, the details about how I enforce that, how I implement it, how I actually put those controls in place for each of my 50 different pieces of software is completely different. It's a different UI. It's a different API. It's even a different model.

So, when we created OPA, it was really designed to solve this problem. To provide a unified way of solving policy and authorization across what we always say, is a cloud native ecosystem. But really, there's nothing all that unique about the cloud native ecosystem. So, it's really about all software at the end of the day. And in order to do that, there are some key things that you'll see embodied in this OPA project that are common across all of these different pieces of software. So, what OPA provides, really, is a – what I'd like to describe as a file format. There's a dedicated file format now that allows you as a user to go in and say, “Here's the logic. Here's the policy that makes authorization decisions.” Now, that logic is domain agnostic. It's not tailored for CI/CD pipelines or applications or Kubernetes. It's a general-purpose policy language, meaning you can apply it to really any of these different, 50 different pieces of software. So, that's like the first thing. It's a file format that says, “Here's how you express policy.” And you can take those as files. Now, I check them into CI/CD do or where want them.

The second big piece was really a policy engine, a piece of software that you could feed those policy files, and then you could ask, “Hey, is Tim authorized to withdraw money from account 1, 2, 3, 4, 5?” And then that policy engine would give you an answer, yes, no, or maybe there's a maybe. Or maybe, yes, he can, but he has to sign this document. Those are all fine decisions. So that's really the second piece. And obviously, it goes hand in hand with the first. The file format doesn't really do you any good if you don't have decent software that knows and understands how to use it. I sort of think of those going hand in hand like a high level with image files, and our browsers can render them right. Like movie formats and how you just open your browser and they start playing the movie, right? It's great.

[00:11:14] JBH: It has one job. It has this one job.

[00:11:16] TH: Does one thing well and then that is policy and authorization, for sure.

[00:11:19] JBH: So, individuals have an easier way to author policy. And then the nice thing is, all these applications simply come out to a policy engine and say, “Hey, run this piece of code”, and you get back an answer if, basically, yes, or no, Tim is allowed, he's not allowed. So, that simplification, then takes all that weight out of the application into this authorization layer. Do I understand that correctly? So, one of the things it kind of blows my mind is just talk a little bit – I think, I don't know if I asked you this question. Maybe I did before. How does this work today? I'm a software developer in a large enterprise. I've got a bunch of applications I'm building. Well, how do I do this today without OPA?

[00:12:01] TH: Yeah. Well, there are a couple of answers to that and I'll call into that. One of which is, like, if I'm an application developer today, and I'm not using OPA, typically what I'll do is I'll write a bunch of Java code here, Ruby code there, Go code over in another service, some React code on the front end. And altogether, those somehow implement your authorization policy. If some auditor comes along and says, “Hey, what is the policy that's implemented?” Or some product owner, some product manager, the answer is, “Well, go look at that bit of React and that bit of Go and that bit of Ruby and that bit of Java and kind of figure it out.” So, that’s kind of the norm, right?

[00:12:38] JBH: When we first talked about this, it kind of blew my mind, because it's one of those things where you just accept it. All these years, I've been working in enterprise software, and I have accepted this idea that all these handshakes require another run of authorization with some other set of rules or language. And then if you have to redeploy, you can break that pretty easily, or have to redo it.

[00:13:00] TH: Yeah. Or heaven forbid you release a new feature now, like what's going to happen? Who knows?

[00:13:06] JBH: Yeah, that's the sort of accepted. I mean, for many years, that's just how it worked. So, that was just really helped me understand the power of OPA and what a great like

simple, elegant approach I thought it was. And then the next question is really helped me understand in a couple of sentences, what is the difference between OPA and Styra's DAS?

[00:13:26] TH: Yup. Right. So Styra DAS is Styra's commercial product. I'll give you two answers. One of which is sort of architectural, and the idea here is that really OPA was designed, and I guess I didn't answer the question earlier. So, I'll do it. OPA is really designed because of how frequent authorization decisions need to be made, right? Every button, click every page view, you need to run OPA as close to the software as you possibly can, physically speaking. If your service is running on a public cloud, you definitely want OPA running in that same public cloud. But not even just the same public cloud. You want to run in the same region, in the same AZ or availability zone.

In fact, ideally, you'd want it running in the same server, maybe even on the same process, as the software that's actually requesting decisions, right? Because what happens if suddenly, like, imagine a reverse, imagine you're running OPA on the other side of the world from your application, or your infrastructure doesn't really matter. And now suddenly, there's a temporary network disconnect between the two. Your software is trying to ask OPA for an authorization decision, and suddenly it can't. It can't get a decision. So, what does your software do? I don't know. Is it going to just let it go? Probably not. It's probably be failing close, which means your software, if it can't talk to OPA, becomes a brick. Now, that's true of any authorization system. And nothing to do with OPA.

So architecturally, what you want is to sort of satisfy two different needs, one of which is you want to run open this case as close to your software as you possibly can, so that it's got high availability on low latency. But you also don't want to embed it into that software, right? You don't want to have that policy littered throughout 57,000 different places in your source code. And so, that's why OPA was designed to run more so as an agent. So, run it as a sidecar. You can even run it as a library. Run it as physically close as you possibly can to that software.

All right, so what that means from an architectural point of view is that you don't have one instance of OPA or 5 or 10, you could have hundreds or thousands of instances of OPA that are all running physically very close to your applications and your infrastructure. So, what that begs the question of is like, "Well, how do you manage? How do you control?" At the end of the day,

what you're trying to accomplish is say, “Well, I would love to have like one conceptual way of writing policy and controlling policy and authorization across different applications and infrastructure. So, you need some sort of control plane, some sort of logically centralized way of managing all those OPAs, very common pattern in the cloud native space, and that's what Styra’s Declarative Authorization Service, or Styra DAS delivers. It is that commercial control plane. It is synergistic with OPA itself.

[00:16:05] JBH: So, what you get from that is running all of these multiple OPAs close to the application, the server, whatever the case may be, and that cuts down on all the kind of, call it hops, right? Network hops, so that you actually get better performance. Is that correct?

[00:16:21] TH: Yeah, it's that, you'd say a Goldilocks zone, right? It's like, if it's too close to your software, then it's intermingled, and you can't separate out what is the authorization policy. What is that logic from the actual app or the infrastructure itself. If it's too far away, then you end up with this poor availability, poor latency, and that turns your software into a brick. So, OPA is intended to be that middle ground where it's architected to be flexible enough where you can, like I said, you can run as a library, you can run as a sidecar, you can run it on the same server as a Daemon. You can go ahead and run as a micro service that’s separate from your applications or your Kubernetes API servers. But the point is that –

[00:17:00] JBH: But that sounds better in theory, probably than reality.

[00:17:04] TH: I mean, sometimes it's the right thing. So, when we chat with folks about what are your policy needs? I always start with like, what's the actual logic that you care about? Where are you hoping to enforce this logic? And then what is the data that your policy needs in order to actually come to conclusions? So, here's a good example, like, go back to the banking app. Suppose that you've got a list of whatever records or accounts, and the authorization logic that you want to put in place as a developer, as a product manager, would be to say, “The only people who can delete a resource or an account or report is the owner of that account or the owner of that report.” Well, who's the owner, right? For every account, for every report in that entire application, you've got to have a bunch of data that says, “This report is owned by Tim or by Jocelyn.”

So, that data is sizable. It is something that OPA needs to understand in some way, shape, or form in order to make a good authorization decision. So, this boils down to like the data gravity problem. So, if you were to try to say, “Hey, I've got a terabyte of data that open these in order to make decisions”, I'm going to go ahead and copy that terabyte of data and to sidecars next to each one of my thousand application microservices, like forget about it. You're not going to do that. So, there are cases, data gravity being a good one, where you actually do want to run OPA as a microservice, and it's not the end of the world, you just have to make sure it's horizontally scalable, and then it's running in the same AZs. It's physically close, like I said, to that app as you can.

[00:18:32] JBH: Takes a little bit more tending. That makes sense. Thank you. One of the questions, so you've got now, hundreds of thousands of these OPAs running in an enterprise environment. The complexity really kind of explodes out in terms of authoring and managing all of these policies in concert. So, you've got your architectural benefits of DAS. Great, got it. Check. But then, I think about these IT leaders and business and product leaders, and there are tens of thousands of policies in every organization, some competing with each other. So, there's kind of like, you said, Goldilocks, but I feel like it's a three wishes scenario. You're like, I wish I had all of this in one control plane. And they're like, “Oh, it's a bit of a gotcha.” Because you see all your policies.

[00:19:16] TH: Yeah, for sure. This is another good dimension. When I'll talk about Styra DAS, I usually say, “It's control plane, for sure.” You need that architecturally. That's a good way of first cut, understanding OPA versus what we do at Styra personally. But then the second level is like, I think about policy lifecycle management. How do you author test, deploy, monitor, record decisions within an enterprise? Especially when you've got multiple teams running into different applications, different infrastructure. How do you make sense of that? How do you put governance controls on that? That's the other second piece that the Styra DAS delivers is, it'll do things like say, “Hey, let's say that you're an enterprise and maybe you want to have – you end up with hundreds of Kubernetes clusters.” Okay. Great. Probably what that means is that you've got different maybe lines of business or different teams responsible for some subset of those clusters, right? Each cluster has its own owner. That team you want to empower to say, “Hey, you put the right policies, the right rules, right regulations in place for your cluster, because you know it. You know what the applications are that are running on it. You know what

your risk tolerances are. You know how sophisticated your end developers are, and how comfortable they are with Kubernetes."

So, we see that happen, like, you've got many clusters, and the organization as a whole, wants to delegate, the administration of all those clusters, to the extent that it can. But it still wants governance in place across all those clusters. So, that's obviously, one of the features that we provide in Styra DAS. The ability to write sort of global policies that apply to many, many, many Kubernetes clusters, or service mesh fleets or whatever the enforcement point happens to be.

[00:20:53] JBH: DAS has special features for that IT manager or business leader who's looking across multiple policies, and it has special features for policy authors. Can you talk a little bit about that?

[00:21:05] TH: Yeah, that's another good one. Another good dimension there. So, when I think about policy authoring, I know that – one of the nice things about OPA is that it gives you this text-based format for describing logic, for describing authorization policy. Some folks love it. They're just like give me VI or Emacs, and I'll crack it open and I'll go to town and write policy that way. Other people will say, "Hey, you know what, I don't want to do that. I want to go and just have you give me a bunch of prebuilt rules. I don't want to write the actual rules at all. I want to go to a GUI and pick one out of a list and put it in place." Some people are sort of in between. They're like, "Well, I don't really want to open a text editor. But I want a GUI to help me guide me through writing policy. But maybe I've got sophisticated policies and I want to write or whatever."

Other people will say, "Hey, you know what, I think about this from a compliance point of view. I care about Mitre, or I care about PCI. So, what I want is a prebuilt wizard that will just walk me through putting a bunch of rules in place that all correspond to things that I know about in those regulations."

[00:22:06] JBH: So like guardrails, just like building in the guardrails ahead of time, so your users can't go off track?

[00:22:12] TH: Right. Yeah. So, the takeaway I always think about is like different personas want to author policy in different ways. They want different amounts of help. So, DAS was

designed to help with all of those that we talked about. You can write it in a text editor. We've got a GUI where you can write text-based policy like an IDE style. We've got a graphical policy builder that you can use, it helps you a little bit more than the IDE. We've got pre-built rules for at least the infrastructure cases where that makes a lot of sense. And then we've got policy packs that cobble together a bunch of rules that I'll help you with PCI or Mitre, or whatever.

So, that's one of the key things that I see in the enterprise, which is that you've got policies when these things that it's a multi stakeholder endeavor, right? It's about security, and developers and product managers, and compliance all coming together and saying, “Here are the policies that we need to put in place.” And I think that's part of why, especially if you think just like littering your code with little bits of policy here and there. It’s so hard, it's because it's not just a developer concern, it's a concern of product and security, and compliance, and so on, and so forth. So, by decoupling those and providing different interfaces for those different personas to interact with that policy in ways that they're most comfortable, does certainly seem to help.

[00:23:25] JBH: I really resonate with what you're saying in terms of policy being a team sport. I think in this new cloud world, where you've really democratized a lot of management of not only your infrastructure, like in the Kubernetes case, but you've democratized access to data, different types of applications, business specific activities. It's an interesting problem of having so many competing policies or policies that are simply changing so fast. The ability – I think that's an interesting kind of part of the velocity of business that's been enabled through cloud, has also kind of amped up this problem statement of do we have the right policy running for today? Also, I think the interesting thing about OPA, too, is to helping develop and create velocity around redeploying new features, right? Because often policy changes will kind of come hand in hand with a new feature.

One of the things I was curious about is, all these features, which seem to resonate as you're working with large customers? I'm really interested in that large, complex customers. Which ones of these features are really the ones that are blowing people's minds or the ones they adopt the most?

[00:24:25] TH: Yeah, let me say one thing. And then if I forget, I'll answer the question directly. I think you hit on something really powerful there, which is there are two kinds of velocity that we think about. One of which is developer velocity, in the sense that especially in the infrastructure case, those rules that people put in place with OPA, with Styra DAS, it may seem odd to say this, but it actually speeds up development. I'll say it again, you put rules in place, to stop things from happening, and it speeds up development.

[00:24:55] JBH: That's right.

[00:24:56] TH: I think it's funny to say it that way, but that same thing happens with traffic signs. In the end of the day, overall, things are working better because there are rules and regulations are governed how like who goes first, and you're not hemming and hawing. I know, when the green light is green, I get to go. And when I get to stop, then I got to stop. But like in the developer world, I think the reason that it goes faster is because developers want to know immediately, if something that they are trying to do is not going to work.

[00:25:20] JBH: Just immediately going to break. It's just immediately going to break.

[00:25:22] TH: Yeah, it's immediately going to break, and so the way I think about the development processes, the sooner you can give developers feedback about like, "No, you can't deploy this ingress this way, or you can't run a random binary from the internet on our production cluster. The sooner they find that out, the better, because then they don't go down a bad path for hours, days, weeks. They figure it out earlier. Moreover, I think one of the cool things is that when you have an automated process in place, it's saying, no, like, not only this is – do you obligate the need for a person to check that. But also, as a developer, I don't care if a machine tells me no a thousand times. I am not going to bat an eye. I'm just going to keep trying. So, it's a good way to learn very quickly and there's not, there's no issues about, "Oh, I got a bad code review, or whatever."

[00:26:10] JBH: It's like lower friction. It's lower friction and less for developers to do. I have to say, every time I see a company that's like its policy as code. And there's a lot of companies kind of using that terminology, my heart always sinks a little bit. Because often, it's sort of like, we're just going to ask the developers to do more things. We're going to add more to the

checklist, right? Whatever it's like, as code is the process that I hear, I'm always like, “Oh, are we just adding more things? More code reviews? More places?” So, in this case, I really do think it helps developers get moving faster, and get over some of these friction points that really should have been automated a long time ago. I love that you're talking about that and double clicking on that, because there's just a lot of companies that are wanting to help developers, and this really does help them in a very obvious pain area, and it helps them in a really efficient, as you said, depersonalized way, which is super handy.

[00:27:03] TH: It just popped into my head, the other – just going back to like this whole notion of policies code and pulling those policies. The reason OPA wasn't even in first place is having this domain agnostic policy, language and file format means that if you've got a policy enforced at the Kubernetes, admission control level, that's great. Why? Because developers can't deploy code they shouldn't. You can take that same policy. Pull it out of Kub now, because it's written in OPA’s language, and enforce it in a CI pipeline. Why is that helpful? Because now, if as a developer, I'm not sending code straight into my Kubernetes cluster, I'm sending it into a CI pipeline first, then it gets deployed, then I can get those checks even earlier, before they get rejected at the Kubernetes cluster. I can take that same policy file and run it on my laptop, so that I don't even have to bother opening a PR in my CI pipeline. Or even worse, waiting until the Kubernetes cluster gets a hold of it. Now, I can basically run these policy checks as if they're like unit tests. I can, in almost real time, I can pull this all the way. We haven't done this yet. But we can pull this all the way into the editor. So, that as you're typing, the Kubernetes manifest or whatever, you're getting real time feedback about, “Oh, you've just put something in there that we're not going to let you deploy.”

[00:28:15] JBH: You're like, “You’re on track. You're on track to succeed. You're falling off track.” Yeah, you get real time. And this is Rego, you're talking about?

[00:28:23] TH: Yeah. Rego, the name of the policy language. Yeah, so you've got that file format. It's just file format. I can send it to you in an email or you can download it out of a Git repo. You can take that file format and run it in an OPA anywhere you like. So, the idea is shift left, that enforcement as far as you can into that developer workflow so that they find out about things that are not going to work as early as they possibly can.

[00:28:47] JBH: I think that's amazing. And Rego, I was curious, when you sort of started the idea around OPA, was Rego always going to be hand in hand with that or is that something you evolved?

[00:28:55] TH: I'll tell you a little story about rego. But the idea behind OPA was really, the way I described it was, we were going to solve that – we're going to provide a unified solution to policy and authorization. That was the mandate. That was the idea. We knew then that we needed a policy language that was not coupled to any of those existing – it couldn't be a Kubernetes thing. It couldn't be a public cloud thing. It had to be somewhat agnostic of all of those. So, those two always went hand in hand. Like the notion of OPA, the only way in my opinion, to solve the problem of unified authorization is to provide that file format, which encodes that language that you use to express those policies.

[00:29:36] JBH: And the reason is because it has to be agnostic as to be some kind of lingua franca for policies specifically. That's helpful. That's interesting. And then do you have any advice for those who are like – if they're getting started writing policies with Rego, is there something different about a declarative language like this, from other languages you might be used to?

[00:29:52] TH: Yeah, what are the things about declarative languages. I mean, maybe one of the main things is that you don't get side effects. So, you don't get to increment counters. It's not a thing you could do. And that seems very basic, but also, it was designed to make the language something that you kind of look at. And ideally, maybe if you enhance some of the punctuation, that you could like read it off in English. This thing is allowed if the user is Tim, and the action is a read, and the resource name is 1, 2, 3, or whatever it is.

So, that was kind of how it's designed that you declare those rules that make up the policy, that make decisions around what you're authorized to do. What you don't then have to do is worry about all the long list of performance things that you would typically need to worry about, and then an imperative language. I mean, I think a lot of imperative languages are complicated by the fact that you can't just write down the logic you care about, right? You write down the logic you care about, that takes an hour, and then you spend 10 hours, making it run quickly. I don't know what the numbers are.

[00:30:51] JBH: I don't know. That's interesting. Yeah, that's right.

[00:30:54] TH: So, one of the premises behind it, right, we would probably argue any declared language, but certainly Rego, was let you make the syntax mirror as close as we reasonably can, English level policies that you might find in legislation. So, that was like priority one, and priority two was like, do whatever we can to automatically deal with the performance optimizations that otherwise, a programmer would have to go off and do.

[00:31:16] JBH: I think that's so thoughtful. That's like such a thoughtful component, that performance forward, taking that off the table really allows this very simple, clear way of writing human readable, declared policies. I just think it's a very interesting component of the DAS and OPA. One of the things I'd love to ask you a little bit about, if you're willing to share is it's a really powerful – DAS is so powerful. You clearly have spent a lot of time thinking about that large, complex enterprise from an architectural perspective, from a user perspective. What's next in the product roadmap? Where would you like to take this product next? To the extent you're thinking about that now, you may just be thinking about tomorrow and all your task for tomorrow. But as you're thinking about that, where's enterprise going, especially these cloud native complex organizations?

[00:32:00] TH: Yeah, well, I think there are a few things to mention here. One of which is we have the –I think we do what a lot of companies will do is at some level we talk to, we've got a bunch of users, or customers, right? They're using thing all the time. They're telling us where they think things need to go. And so, we're doing that. But what we also have is the open source community. We're watching what they're doing. We're chatting with them, helping them along. And as we see trends show up there, then we go ahead and think, "Hey, there was recently a CloudFormation, or AWS release to hook into CloudFormation." And then we did in the OPA world, we put together some work that says, "Here's how you hook OPA in to CloudFormation, so that you can put policy guardrails in place for AWS, which is great."

Very similar to how we do with HashiCorp, right? We just recently talked about and released this work in Styra DAS around guardrails for TerraForm. We had that for a while, but just recently, HashiCorp have opened up TerraForm Cloud, to have that same kind of hook so that we can go

ahead in there. So, sometimes what we do is we just watch what other vendors are doing. And we say, “Hey, you know what, that's a powerful place for us to integrate OPA.” And then we can go ahead and provide that and a bunch of the stuff around that and bake it into DAS. So, those are two of the things that we're excited by.

But then there's also, we just announced this Styra Run product where for the first time, we're actually running OPA for folks in the cloud. So that's pretty exciting. Up to now, we've had that very clear separation between OPA is the open source, you run it at the edge, and Styra DAS has been that control plane. And now the Styra Run product is really designed to be able, we’ll run OPA for you in the cloud as a managed service. And then it makes, the idea being it's easy to offload your application authorization decisions to Styra Run.

[00:33:44] JBH: Any technical surprises as you've moved your service into a fully cloud based services?

[00:33:49] TH: Well, Styra DAS has always been fully cloud, run in the cloud. The interesting bit about Styra Run has just been the things that we always knew, right? That you need geo replication, you need really low latency, you need highly optimized memory, and tenant and sharing. So, I don't think we've run into anything that was surprising as we've been working through this, but –

[00:34:12] JBH: Just the regular slog of things that known items. The regular known problems.

[00:34:18] TH: Right. This is something that obviously, we've been talking with people about for a long time for years. So, I think, once we decided to do it, and we felt like we had a pretty good handle on what was going to be required.

[00:34:29] JBH: I want to double click on two ideas here that you just checked on. One, is I had a question about integration, and one is about leverage and your open source routes. So, let's talk a little bit about integration patterns. Because you're right, the part I see the most is the policy authoring and the dependency on the input data, which is all like immaculate and perfectly organized, which never is, right? So, just help me understand, if a big customer signs

up, like how do you start these integrations and implementations, and what are some of the keys of success for an implementation for you?

[00:35:03] TH: Frist, comments. So overall, like one of the powerful things about this cloud native ecosystem is it just architects really, we're seeing this move toward authorization being basically being a hook that's available in cloud native software. So, you can plug things in like OPA to it, right? Kubernetes does this. They've got a great pipeline for authentication and authorization for admission control. So, when you're running OPA, you're hooking OPA into that API server. There's no way to circumvent it. We're not running a proxy or something in front. You don't need to. And I think that architecturally that's becoming a pattern that we see within the cloud native ecosystem. Service meshes do this. We've just gone through Hashi and AWS doing this. So, we're starting to see databases do this, too. So, there's more and more of these software components. They're just making authorization a pluggable feature. And that's obviously powerful for OPA and Styra.

[00:35:52] JBH: Yeah, it makes life a lot easier. That's great.

[00:35:54] TH: Yeah. And so, when we're working with enterprises, often what we'll say, we'll tell the story that we've been telling here today, which is, “Hey, there's authorization everywhere, within the applications, within the infrastructure. It's just everywhere and we help you with all of it.” And people love that, right? Because why wouldn't you? But at the end of the day, to make progress, what we always say and shot this from the rooftops, pick a pain point, pick that application where you already know that you're in pain, and you need to factor the authorization logic out of it, or pick that infrastructure, technology, whether it's Kubernetes, or TerraForm, or whatever, where you know you need to put guardrails in place. Go find that team that owns Kubernetes, that owns that application and understand directly. What are your pain points? Does decoupling authorization from your application or putting guardrails in place for your infrastructure? Are those things that will help you address those pain points? And if so, then roll it out.

So, you get one win that way, it doesn't matter what the area is, and then you've learned a lot about how to use OPA. You've learned a lot about how to use Styra DAS. You've got a great story to tell and explain, like, here's how we got this to work in my application one, or in my

Kubernetes for the organization. And then you go and you say, “Okay, what's the next one on the list?” And then you repeat. That's the way I think to go from. So, it’s kind of like a bottom up kind of approach to getting authorization controls in place everywhere. If you're solving concrete pain points, then nobody's going to stop you. If you're trying instead to say, “Hey, as an organization, everyone's going to do it this way”, that's going to be a very, very long road.

[00:37:32] JBH: I love that as a product manager and a technology person. That kind of really discreet use case as the starting point. For me, I call it the race car strategy, you just pick one car to beat, and then you pick the next car to beat until you're at the front of the pack. Because I do think like these efforts to create some kind of master set of rules, always just kind of fall down under their own weight.

That's actually a great kind of segue into talking a little bit about the business strategy behind Styra DAS. I did have say that little bit of a tease there. You said there was a good story about when you were thinking about starting this effort and building OPA, and you were thinking about Rego. Yeah. So, tell me a little bit about that sort of origin story. You're there. You've got your pizza box. You've got your partners. Right?

[00:38:20] TH: Yeah. All right, I'll do the origin story. The thing about Rego is a cute little technical anecdote. So, if we have time, remind me and I'll do that one.

[00:38:28] JBH: I want to hear that right now. That's what this whole show is about, technical anecdotes.

[00:38:34] TH: I always start with the origin, and then we go. So, before we started company, we were at VMware. We had ended up at VMware through the Nicira acquisition. And it's funny in the early days in Nicira, it was really a policy networking company. That was the idea. It was like, let's write some policies that govern where packets can flow and where they can't. And then from there, what came out of it was virtual networking, just because it was easier for people to consume and to write. And then, obviously, it went off to software defined networking.

So, while we're at VMware, some of the clients we're talking to knew that we were some policy folks. And they were like, “Hey, by the way, we're a bank, or we're a tech firm, and we've built

these policies, systems to manage all our policies, and why are we doing this? We don't want to do this. Hey, can you guys like go off and do something." We'll give it a whirl. We don't want to be in the business of building and maintaining a policy system. That's not what we should be doing.

So, we spent a couple of years at that point, building out a project inside of OpenStack, at the time. This was nine years ago at this point. So, it was well back then. We went into OpenStack and we built it. It's called OpenStack Congress, and we spent some time on that. And then what we decided was, look, this policy problem is really bigger than OpenStack. It's bigger than VMware. Every company on the planet is going to have this problem, especially taking into account the fact that all these companies are embracing this cloud based, these cloud native approaches to building software. And when you do that, you've got way more software components, right? Like you're talking about going from monoliths to microservices. You're talking about far more dynamic environments. You've got Kubernetes spinning up and down instances of your microservices all the time. You've got incredible reach in terms of where you're deploying all of these software application instances.

[00:40:20] JBH: So, you're at VMware, you're building this, you've got background, and the whole policy world that you've been thinking about, and then from a network perspective, for a long time. When you thought about open sourcing this, tell me a little bit about how that strategy came into play?

[00:40:33] TH: That's a good question. For me, there was never a doubt that we had to open source the policy language that became OPA, right? For the simple reason that policy is so crucial to so many different parts of the business. Like I'm imagining going – imagine you're a 10-person startup, and you go to some – the largest bank on the planet and you're like, "Hey, you want to write all your policies in this proprietary language that only we have control over? Is that going to work?" No. I just couldn't bring myself to believe that would ever work.

[00:41:04] JBH: You know what I'm really interested in knowing is, so you're like, I love it, right? I'm all about open sourcing. You're going to open source this. Of course, it makes perfect sense for your business model. Did you know kind of the shape of DAS at the same time? Because it's such a full featured enterprise offering. A lot of times open source, I see like a great idea, and

then they're kind of struggling for that enterprise product layer. This seems much more heavily weighted towards that enterprise product. Was that part of the plan from the beginning?

[00:41:30] TH: Yeah, I'll just finish the thought of it. I did half of it, which was we couldn't imagine the company succeeding if we hadn't open sourced OPA. But the other side of that was, that if we open sourced OPA, we'd have everybody in the world with expertise in all kinds of different domains contributing to the language, contributing to the engine. Bringing use cases that we wouldn't have envisioned or thought of. I saw it both ways is that, like, that is a necessary component of not only necessary, but it's wildly beneficial to have this thing owned by the world, owned by the community.

Okay, yeah, was DAS in there from the beginning? The answer is yes, conceptually. So, the other part of the backstory here is Teemu, one of the founders, was the Chief Architect at Nicira. So, Nicira was this distributed network, software defined networking. They had the same kind of problem that we did, which was they needed this piece to run at the edge, because turns out, networking needs high availability, high performance as well. But they also needed that centralized control. So, we always knew coming into the authorization space, that that was the architecture overall that we needed. We needed this data plane, as the networking folks like to call it, that runs at the edge, that's OPA, and it became OPA. And then we needed that logically centralized way of controlling all those OPAs. So, that was, in part why, Teemu and I got together was that we knew that that was the right technical architecture, to solve authorization from the very beginning.

[00:42:59] JBH: That's really helpful because it really is a mature offering from an enterprise perspective. And of course, its integration. Its handshakes with OPA are so clean. So, that makes perfect sense and I hope other young founders who are thinking about an open source endeavor, I think it's one of the things that's interesting about what you're saying is you were thinking about the enterprise from the beginning, and that doesn't make you any less virtuous as an open source leader to have that in mind, right, ahead of time. So, one of the questions I had for you as a leader in open source is how should investors, how should buyers, when they're evaluating the health – a lot of times I get asked, well, how do I know if it's a good open source project underneath that enterprise offering? So, a lot of commentaries out there, but I'd love to

get your opinion on what do you look for when you're like, “Oh, let me check out and see if this is a solid project.” What are the kinds of things you look at?

[00:43:49] TH: Yeah, well, there are a couple of good signs that it's a healthy project. One of which is that you've got a broad contributor base. I don't know if I'll say anything uniquely interesting here, but broad contributor base, right? That's active. So, you're seeing contributions go in, you're seeing bug fixes go in, you're seeing new features released, you're seeing a good release cadence. That's on the sort of developer side. If I think about users, what I always think is I want to hear not from the people who built it, how they think it should be used. I want to hear from the users, how they were actually using it. Were there rough edges? Where did it work really well? Where did it sing? So, like looking for end users who are active in the community who or maybe they're on Slack, and you can ask them, or like for us, we always make sure that we encourage people to give talks at like Kub Con, or any other sort of open source summit, so they can share what they've learned. If you've got folks who are leading lights in the industry, giving talks at premier conferences about getting detailed descriptions of how they use an open source project and why it's been successful, and then I think the project is probably quite healthy.

[00:44:56] JBH: Is there like a cultural aspect to your open source project? Some are crankier than others. Some are more fraught than others. How do you kind of set the culture for your open source community?

[00:45:07] TH: Culture is a good one. I mean, I think it's – well, I mean, it goes without saying. It's super crucial for any group of folks, right? Whether it's open source or commercial or a company or a collaborative thing. In terms of setting the culture, I'm a believer in you got to walk the walk, right? So, how you engage with your community, other people are going to copy, especially for the founders, the maintainers, the creators. So, what you want to do, certainly if you create it and maintain the project is, you should be on Slack. Maybe not every day. Maybe you can't do that. But you should be on Slack and interacting with folks. You should be them. You should be thinking about doing that customer success function in the enterprise world. That should be open source team on Slack or Discord, or whatever it is.

So, part of your job in open source is not just to write code. I think healthy open source projects, especially, have those core teams that are writing code for sure. But they're also writing docs. They're also interacting and answering questions on Slack or Discord. They're giving talks in community forums, going to meetups even, right? Obviously, there's a scaling question there that you can do less and less of that.

[00:46:16] JBH: Yeah, that's kind of what I'm getting at, like, you have these leaders who get very focused on these technical problem sets, which are very delightful, I get that. But then there's this problem of bringing people into the fold and keeping that healthy, like influx of new users, new contributors, which I think is kind of an interesting tension, you've got these super technical people who want to go deep on the tech. But then if you keep doing that, then you end up with not enough new people coming contributors. So, it's interesting, and I think it's just such a lively project. It's something to be really excited about.

I'm just curious, I work with a lot of MBA students here and there, and there's a lot of hand wringing about whether they should go to a startup, or go to a large company. I was just curious, I'd love to be able to tell people that there's benefit to being in a big company, when you come as a founder, it's not a detriment to have that big company background. So, I guess my question is, what did you find that you brought with you from the large company experience?

[00:47:14] TH: Yeah, at the end of the day, when you're a startup, you're going to have to sell to other companies. If you've only ever worked in a startup, it's going to be very hard to imagine what it's like in a big company and that's who you're often going to sell to. If you're selling to other startups, okay, that's great. But what 90% of startups fail, or 99%, or whatever that number is. So, that's helpful. But I think you're living in a small startup so you can imagine what that is like. It's hard to imagine what it's like in a big company unless you've already worked at one.

[00:47:41] JBH: Interesting.

[00:47:42] TH: So, that would be one thing, just like that basic knowledge of like, “Yes, we've spent two years designing a new product, and it didn't get released.” Or whatever. And yes, I have to go through get three levels of approval to purchase this piece of software. I really can't even do that. I had to go get my manager. That kind of socialization, that kind of –

[00:48:02] JBH: You're less dismayed. You have less dismay when you are confronted with this timeline.

[00:48:09] TH: It's just like empathy for your users, for your customers at the end of the day, and I think it's just very hard to understand that. Now, that's like just very vague and kind of squishy. But I think the biggest difference that like somebody who's worried about like, should I go into a big company or a small company first? If they eventually want to found a company, is just that like, I think you do want experience especially in – working in a big company is great, if you can work in the area that's somewhat related to what's you're going to go find a company around. And obviously, that's hard to know. But like, if you spend a bunch of time in marketing and a big company, then when you leave, it'd be great if you could sell a marketing product because you know what that is like, and what procurement looks like. What it looks like to bring in a new piece of software and train everybody up, and so on and so forth.

[00:48:56] JBH: I love that, because you had this theme of policy in your regular work life, and then it kind of kept your thought process going and you had a domain to extend into. That's actually great. I'm going to use that. Tim, I really appreciate your generosity to come on the show. Great comments. I've learned a lot about Styra and I look forward to talking more in the future.

[00:49:16] TH: Thank you so much for having me, Jocelyn. This was fun. As always, I'm always happy to come back.

[00:49:20] JBH: Great. Thanks so much.

[END]