EPISODE 1195

[INTRODUCTION]

[00:00:00] JM: Security is as important as ever especially in regulated fields such as healthcare and financial services. Developers working in highly regulated industries often spend considerable time building tooling to help improve compliance and pass security audits. While the core of many security workflows is similar, each industry and each organization may have its own idiosyncratic needs or particular regulatory requirements to meet.

Sym is a platform for building security workflows that seeks to build on those core similarities while empowering developers with the tools they need to meet their application's unique security and compliance needs. Sym believes in putting engineers in control of security in the same way that DevOps puts engineers in control of infrastructure.

Yasyf Mohamedali is a CEO and co-founder of SymOps. Before SymOps, he was the CTO of Karuna Health. He joins the show today to talk about security and innovation in regulated industries and how Sym can help developers close the intent to implementation gap in application security.

[INTERVIEW]

[00:01:10] JM: Yasyf, welcome to the show.

[00:01:12] YM: Thank you so much.

[00:01:13] JM: Describe the engineering problems that exist in heavily regulated spaces.

[00:01:19] YM: Yeah. Great, a great way to kick us off. I think the really interesting challenge you see in these heavily regulated spaces today is it's less of an engineering challenge and more of a tricky tradeoff, I'd say. There's like a lot of engineers out there who are really keen on doing the right thing by security and compliance, but understandably these are also engineers who are on teams where you have to move really fast and pump out a lot of product. And so the kind of tradeoff you often end up facing around security compliance is basically move faster, be secure, and that's basically the choice you get almost every time. And so that's actually something that we're really excited about tapping at Sym is kind of limiting that tradeoff. But I'd say that constant tension of do we slow down and do the right thing or do we meet our product deadlines is the biggest challenge I see in the space today.

[00:02:07] JM: And can you give me an example of a heavily regulated industry and a typical problem they might have?

[00:02:13] YM: Yeah. I think let me give two very different examples. So one is kind of more what you'd expect when I say highly regulated industry. So maybe something like a healthcare company where let's say it's a SaaS company that happens to be in healthcare and they're building a tool. I'll use my last company as an example. We were building a messaging kind of product for healthcare. And messaging products have all sorts of really cool technical problems, and all we want to do is like solve the technical problems and use some like – We did a bunch of SMS and callings. We want to be able to use Twilio or some other vendor, kind of like solve the underlying APIs for us and we could just do the cool stuff of like how do we compose these things and build a product out of this.

But being in healthcare, we had all of these kind of hoops you had to jump through in order to use third-party vendors and all this kind of internal kind of like software and tooling we had to build to make sure that health information didn't get leaked to these vendors and to do kind of quarterly reviews of all of our infrastructure. And a lot of this kind of almost like governance kind of tools where every month you'd be like, “Well, here's the stuff we could get done.” But instead here's like the three out of five things will actually get done because we have to spend two of the five slots building uh a tool for adding like peer review of SQL queries or something like that.

And so that's the kind of more classic example. I think a more very specific example that's more topical people is look at the Twitter hack last year, right? The Twitter hack was effectively internal, kind of admin dashboard that didn't have any kind of like multi-factor on it. So someone compromised access to this dashboard, they had access to everything. You can bet that the engineers who are building it at some point someone thought like, “Oh man! We should put some kind of MFA or a two-factor on this dashboard. It shouldn't just be like you log in with your static credentials and you get access.” But then they have to decide, “Do we go do that or do we meet our sprint goals for the next three things we have to build?” And they chose understandably to go meet their sprint goals for the next stuff and the rest is history.

[00:04:13] JM: So these engineering problems, they're typically getting addressed by just internal engineers writing custom code to fix them, right? Like is there a generalizable solution to these kinds of problems that you're discussing?

[00:04:30] YM: Yeah. I think right now there're three different solutions I see people going after, and none of them are great. So the first is the kind of obvious one of, yeah, let's just write a bunch of custom code. Have the internal tools team kind of whip up a solution to this, what is effectively often a workflow problem. And it's super time consuming and slows you down as per the tradeoff we just discussed, but also there's this huge maintenance burden you have to worry about, right? Like building internal tools, whether it's like a ChatOps bot or whether it's a microservice that's like auditing things or issuing credentials or whatever it is. You don't have to worry about security of that thing, vulnerability scanning, pen testing, who's going to maintain it? The list goes on and on. So building this internally is like often an obvious solution for engineers, but actually has this huge hidden cost.

So then people move to like one of two other solutions. One is like buying off-the-shelf like security products that solve your one-off thing. And often those are older products that are kind of meant for an IT world, right? Where it's like it provisioning security software. And the problem with those tools is they’re very inflexible, right? Anyone has ever had that moment where you just like wish like this website or product had an API so you just customize it and do what you want but it doesn't? So you're stuck kind of building like human workflow hacks around this inflexible product? That's where you often end up with like buying things off-the-shelf, which then takes people to option three, which is like we need more flexibility. We don't want to build something, and so the best way to compensate is humans.

And so really, unfortunately, the way this problem is solved ultimately a lot of companies is let's spin up a juror ticket queue. Let's put an ops person on the other end of it, and whatever we need, this human will just do, which is terribly inefficient. Not at all what that person signed up to do for their job, but ends up being kind of where most companies land because they don't have a toolkit to do anything else.

[00:06:15] JM: Take me inside the engineering companies that we're talking about here. Like what are the problems that they're facing in a little bit more detail? And how are they evaluating these problems? What infrastructure assets do they need to secure to solve these problems consistently?

[00:06:33] YM: Yeah. So I'd say there's a few big kind of like specific problems these companies are facing. And when I say these companies, I really do mean at this point anyone is building software. Everything from small startups, to big tech companies, to large enterprises that happen to employ a small number of engineers. And I think that the big buckets that at least we see a lot, one is this notion of just giving kind of ephemeral access to things. So like I'm an engineer. There's an S3 bucket that has sensitive logs in it. I don't need access to that bucket normally, but like I'm on-call. I got paged three in the morning and I need to go look at those logs. How does someone give me access but make sure that they're not just like flicking a switch that never gets turned off, right? That's like good tooling around ephemeral and like just-in-time access is one.

Another big area we see is kind of like onboarding, offboarding, right? So just like the crap of like, “Oh my God! Let me spend the next three weeks now that I'm onboarded figuring out all the services I don't have access to you and like hunting down the right person to give me access.” And likewise that horrible feeling when you realize that someone left your team two months ago and they still have access to your production AWS account. So I think that's the second big bucket we see.

And the third is this kind of long tail of like taking everything around security including the two things we just talked about and kind of thinking about the compliance end of those things, which is like how do we demonstrate that we are actually following these workflows and doing the right thing? And that becomes a whole other issue where everything's internal or external audit. There's this last minute mad dash. People are like pulling records and scraping their logs and running SQL queries and it's just like super stressful time because there's nothing baked into the tools from the get-go that's actually kind of recording and logging all these kind of audit trails. And so you end up with this kind of like huge last minute mad dash that kind of compounds the other two problems we talked about.

[00:08:25] JM: You're building Sym to address some of these problems. Tell me about what you're building and how you're addressing them.

[00:08:32] YM: Yeah, totally. So our goal – We kind of took a look at the ecosystem and we had two insights. One was kind of the thing we talked about earlier, which is when someone needs to go implement new security workflow, their options right now are write a bunch of difficult to maintain code. Use an inflexible product off-the-shelf or just get a human to do it. And none of those seemed great to us. So we started to kind of build a developer tool to help with this. And really what we wanted to build is kind of workflow automation platform where you could kind of codify these workflows really easily and bring some automation but without bringing a large maintenance burden and also while still maintaining kind of – Being in a place where the workflows you end up with are as unobtrusive for your engineering team as they would be if you had built them from scratch yourself.

And the way we went about doing that is our second observation in the market, which is that if you look across companies, compliance standards, industries. What you find is that the security workflows people are rolling out are actually quite similar even across very different companies. And what you normally see is people are building the same kind of core primitives and they're kind of arranging those primitives in a different way or putting a different kind of skin around it, and that skin is normally connecting to external systems that are already in place. And so that's kind of exactly what our product is matching, that second observation.

So what Sym's product is, is its infrastructure as code which lets you provision these core kind of workflow templates, which are almost these primitives that everyone ends up building over and over. And then once you provision those primitives, we have an SDK that lets you pull down a long tail of integrations and wire them up with your template to kind of put that last mile kind of customization on your workflow such that it looks exactly like whatever you would have built in-house, but you kind of used our building blocks instead of writing all the code from scratch.

[00:10:27] JM: So the product that you're offering people is the templates for solutions together with the integrations that you've built. Is that right?

[00:10:39] YM: Correct. Correct. So, very concretely, it's a Terraform provider that connects with our cloud hosted solution, which can also be deployed on-prem and lets you provision these workflow templates. And then it's a Python SDK that lets you kind of write you know a few lines of imperative code to customize these workflow templates and kind of craft the exact workflow that meets your needs.

[00:11:02] JM: Can you say more about what a Terraform template is and what problem it solves?

[00:11:07] YM: Yeah. So think about it. So like I said, we're building these kind of workflow templates, which are kind of these primitives. And the way we think about those is those are a piece of infrastructure like anything else. So these days, the popular thing to do when you have some infrastructure is you use something like Cloud Formation or Terraform to kind of declaratively provision it. So you say, "I want an S3 bucket that has this name and this access policy." Or you say, "I want this database that has these things."

And so, similarly, we want people to be able to say, “I want this workflow template that has this logic.” And so the way we do that is we give them kind of what's called a Terraform provider, which kind of plugs into Terraform and offers a custom kind of piece of infrastructure that they can provision declaratively. So they can say, “I want a workflow template that is called this and that inherits some of this logic and they can run Terrorform, apply on that kind of static configuration and get exactly what they want.”

And lots of companies out there are using Terraform. And lots are not using Terraform. Lots are not using Terraform. Our product is not necessarily bound to just people who use Terraform. It's just a tool we use for letting people provision infrastructure. And it could be any other kind of infrastructure as code tool.

[00:12:22] JM: So what is an example of a template that somebody might want to deploy?

[00:12:29] YM: Yeah. So the template that people are most commonly using today with Sym is called the Sym approval template. And basically what that approval template is, is it sets up –You can almost think about templates as like in the Docker world. You have kind of these like parent Docker images that lay out like a base file system. A Sym template almost lays out like a base set of workflow steps. And the Sym approval template lays out the base set of workflow steps for granting someone approval to some resource. Routing that request to the right people. And then once it's been approved, kind of actually making the change to let the person into whatever the resource is as well as kind of removing them from the resource when that access is expired. So I can make that very concrete, because that's a bit abstract. Two concrete use cases from the approval template, one is a very classic like I have an S3 bucket. It has sensitive logs. And I want to say that any engineer can request access that bucket. When they request access, we follow some kind of routing algorithm. Maybe we say if the engineer is on-call, they get auto approved. If they're not on-call but they're a manager, then any other manager can approve. And if they're not on-call and not a manager, then their own manager has to approve the request. So maybe that's what the logic looks like and you can write all that in like three or four lines of like Python in our SDK. But like that's an example of what this workflow could look like, is an engineer requests access to S3 bucket. It follows that kind of complex routing logic. And when the right person clicks approve, our system automatically gives them access to the bucket. They go in. Everything they do is kind of monitored and logged. And then when they're finished, the access is automatically revoked. That's one example of a use case on this kind of Sym approval template.

[00:14:15] JM: The building blocks that you've created to make these kinds of templates, can you go into a little more detail of what these building blocks are? Like what you've actually built as a company?

[00:14:26] YM: Yeah. So the building blocks are basically the steps that comprise these templates, and basically – So myself and both co-founders have all kind of run engineering teams in kind of heavily regulated environments before. So you basically just took the set of tools, the block that lets someone submit an access request from a command line or Slack, or the block that lets someone write a function that defines where request gets routed, or the block that you know gives someone access to something after a certain amount of time revokes that access. These are all like very common chunks of code that people write over and over at different companies. So we kind of wrote all those blocks once in this kind of generic way and then put this kind of layer of these kind of parameterized templates on top of them.

And so it's kind of cool, because when people go provision one of these templates as a piece of infrastructure, the resulting workflow they get has all of this like battle tested logic that is kind of the ultimate form where they would have built internally, but shared across all the companies that use Sym.

And likewise, on the kind of integrations and SDK side where people are writing some Python to customize those workflow templates, we have this large library of like imports they can use that connect to various different services. And we like to say that like every one line of code you write in Sym is probably two or three lines of coding outside of Sym, because we've built these nice abstractions and tools and imports for you where you'd otherwise have to go spin up a whole microservice or write a whole integration with an external API. So, really, it's taking all the stuff that gets built. Building it once and then exposing it in a really developer friendly way so you can like string things together in an order of kind of minutes instead of weeks.

[00:16:13] JM: And can you say a little bit more about the deployment and integration process for the average customer?

[00:16:21] YM: Yeah. It's pretty cool model actually. So there's two components to the Sym platform. One is a cloud-hosted control plane, and that's – We host that and it's kind of – It's really command and control for the other part of the Sym product, which is this virtual appliance. And what's cool is this virtual appliance is the thing that runs all the user kind of SDK code. It's the thing that does all the sensitive operations. It's the thing that has all of the credentials. And that virtual appliance, it can live in Sym's cloud, but it could also be deployed to your own environments. So you can deploy it to your AWS account or to a GCP account.

And so the really cool thing there is you don't have to trust us with any of the sensitive credentials or even in your workflow logic. You can use custom code. You can connect to any external or internal service and all of that is executed within your own environment. So in that sense, you can kind of use Sym for these like very sensitive security operations, but not have to worry about like, "Oh man! They're getting the keys to our kingdom."

[00:17:24] JM: Gotcha. And I think it's worth just – Because people may have lost the thread of what we're talking about here, let's kind of like think about a use case. So like something in healthcare or something at the – Yeah, let's go something in healthcare, because you've been the CTO of a healthcare company. What kinds of problems would Sym have solved for you back at that company? And maybe dive a little bit deeper into how you would have used it? How the implementation would have worked?

[00:17:52] YM: Yeah, yeah. So let me give two very concrete examples there. So one example, we had a customer that came to us and kind of – I'll make sort of a hypothetical situation. Hypothetical customer that came to us and kind of said, "Hey. So we just want to make sure, because we're giving all this patient data, we have to all comply with HIPAA. And one of the parts of HIPAA that's kind of specified for engineers is this thing called the minimum necessary requirement," which basically says only the exact people at any given time who need to have access to sensitive patient data have access to it.

And so this customer is kind of auditing us. They sign a big contract and they're saying, "We want to make sure that you're following the minimum necessary requirement." And so they're kind of doing an audit of us and they're looking around. They're saying, "Hey, we notice that you have this database and it keeps all the patient records. But you told us that there are eight people that have access to this database and we notice the size of your engineering team is eight. So why does everyone have access to our patient records that doesn't sound like minimum necessary requirement?"

And answer to that was like, "Well, we want to keep the service as reliable as possible for you. Sometimes when things come up last minute, engineers need to go take a look at that database and like pull out records to use for debugging." And the customer's perspective is like, "Well, that's great when that happens, but the rest of the time any one of your engineers would be compromised and now all of our patient records are out there in the wild."

And so they said is, "We want you to put a workflow in place that ensures that engineers can go in and do their job and save us from downtime when they need to. But on the average day, they're not able to just go and look at all these records." And so our choices were, "Okay, we can look at some like specific like security solution for putting this like requesting and granting access around a database, but I don't even know where to start there. We could write a bunch of code to like write some scripts and stuff internally ourselves, which is actually what we end up doing. Or we could just hire like a like an ops person and say like, "Hey, engineer, whenever you need access to the database, like go wake up the ops person and like submit a ticket and let them go give you the access and leave a sticky note on their desk to like revoke the access after a couple days," right? And like we've all seen that before at companies and we know why that kind of sucks, right? Not fun for the ops person being a gatekeeper and not fun for the engineer either.

So then you can turn your attention to like, “Okay, well let's say we have the time to go build our own workflow around this like. What would we build?” Well, probably I'd want to be able to like request access and the tools I already use. Maybe like my terminal or like my Slack. Probably, I'd want to be able to like very easily configure where those requests get routed to, but I want it to be like different routing if it was an emergency or late at night. And probably I want automation of like actually granting my access and working that access.

So we built that at my last company, and it took us a month and we thought, “Great! We spent a month and we’re able to meet the customer requirement and we closed a big contract.” But if Sym had existed, we would have been able to pull down a template, provision it, write a couple lines of Python and have the exact same workflow that fits perfectly into our existing tools that we've already built ourselves, but done in a matter of minutes or hours instead of weeks.” So that's one very concrete example.

And a nice kind of extension to that is we had a similar issue after that where it's like, “Okay, there's another database that doesn't have sensitive credentials necessarily or sensitive information necessarily, but it's got like core data that's necessary for production and we had an incident where one of the engineers ran a bad query and almost dropped the database.” And doing a postmortem for the resulting downtime, of course, the first thing customers want to know is like, “Well, what are you doing to make sure that people can't do this in the future?”

And, again, it's like our options before would have been like just revoke everyone's access and put some like crazy manual ticket-based process in place to be able to run these queries or buy something off-the-shelf that's going to like do some funky machine learning or something and like analyze queries. Like that sounds too complicated. Or do nothing and risk losing a customer. Whereas like, really, again, if we had had time, we probably would have built internally a really kind of cool homegrown workflow where someone can kind of like make a query on a command line. And if that query is part of a predetermined set of queries that are safe to run, it just runs. And if not, like the manager gets like a push notification on their phone saying like, “Hey, Yasyf wants to run this query. Just tap yes to let them do it.”

And so we started building something like that, because that was useful for us, but there were so many pitfalls to building a complex system like that. Again, if we'd been using Sym, rolling out that workflow where we kind of wrap a database in this kind of like audited kind of queries and roll in different rules for who can approve those queries and how they can be pre-approved and how they get routed is all at our fingertips within a few lines of code.

And so in general, there're some platforms kind of this thing where any time there's this kind of just-in-time access workflow for any kind of resource, whether it's something in your cloud, whether it's a capability, like running a SQL query, or even whether it's some application level thing like the admin dashboard we're talking about for Twitter. Putting a very custom workflow in front of that where at the core someone is requesting access, someone is approving access after being routed that request, and there's some kind of automatic yes, let you do the thing with an optional let you stop doing the thing later, is a handful of lines of code no matter how you spin it on the Sym platform, whereas each one of those workflows would have been several months of work without the Sym platform. And all of that is just one example of one of the templates offered on the platform. So it gets pretty powerful when you start thinking of all the different things you can do.

[00:23:46] JM: Is it at all difficult to explain what you are? How you do? How you do what you do? Just because it's not like a SaaS platform that's just kind of easy to explain. It's this set of tools, the set of building blocks that is kind of made available to developers. Is it hard to explain?

[00:24:07] YM: Yeah. It's something we're slowly getting better at. But as you've seen even here, it's kind of difficult. I think the high level of like being able to tell people like, “Hey, at companies that employ a lot of engineers, there are security things that come up.” And when those security things that come up, when those security things come up, the engineers have three options. Do nothing, which is a big risk. Build a solution themselves, which is very resource consuming. Or buy an off-the-shelf solution, which slows them down because it's obtrusive. Like people kind of understand like, “Okay, I see why all three of those options suck.” And then being able to say, “Hey, we're option four. We let them kind of have the workflow they want without –" So if they don't do nothing, they don't have to spend a ton of time and money and they get the flexibility they want so they don't slow down. So we're better than all three of those options. That kind of high level of like we let engineers do that and like let's just wave our hands about how we let them do that. People kind of grok that.

But to dive a level deeper and talk about templates and workflows and integrations and Terraform, it gets pretty dizzying pretty fast. I think that the silver lining is when you talk to engineers who are in the field, DevOps and SRE people who are living and breathing this every day, so people that are customers, they kind of get it right away cause I'll start telling one of these stories and they'll be like, "Oh, that happened to me last week," right? And so the nice thing about our customers is they pick up on it right away and they kind of really understand the value of the platform.

[00:25:36] JM: Can we go through another security workflow or just explain a few more security workflows to kind of emphasize the generalizability of what we're talking about here and how many security workflows the average company might need to implement?

[00:25:54] YM: Yeah. So let me give two more examples. These are both – Well, the first one actually, a lot of companies now these days end up adopting pretty early on and it's kind of doing some kind of what is normally a quarterly risk assessment. So the way this works is for a lot of different compliance standards, a lot of different kind of audits, on some basis whereas annual, monthly, quarterly, you kind of have to take stock of all of your infrastructure and resources and say, "For each kind of set of infrastructure, let me do this assessment which says, "What are the different ways this thing can fail?" Whether that's failed because it does something bad like leak data or fail because it goes down.

And for each of those ways it can fail, how likely is that and how costly would that be as a business? And you kind of like multiply all the probabilities through and you end up with these kind of numbers that kind of tell you, give you kind of a risk score for each class of assets around your company. And a class of assets could be a set of databases or some S3 buckets or some servers or even a third-party platform.

And so this kind of, what's called a risk assessment, it's pretty commonly done at a lot of tech companies now that have to comply with standards like SOC 2 or HIPAA or PCI or any of these different kind of industry-specific standards. And so right now it's just like terrible process where like once a quarter, 10 people get in the boardroom and spend like eight hours there and just like run through this like God awful spreadsheet. Anyone who's listening who's gone through this right now is probably smiling, because they've been there in that room, and it’s terrible.

At my last company, we would like order pizzas and like I'll try to hang out and like make a fun day of it, but it was awful. And if you think about it, on the spectrum of like the problem Sym solving, remember, one extreme was like do nothing. Or I guess three options. One was do nothing. Two was do it manually. And three was buy an inflexible product. This is kind of option two. People are doing it manually. They're not using any automation. They're just doing it all themselves.

And so when you think about what an automated kind of like ideal workflow would look like for a process like this, like it would be like totally automatic, right? The system itself knows what the risk of every asset is, and anytime you change one of those assets you get a notification saying, “Hey, you changed this thing. Please just let me know what are the updated risk numbers for with this change you made.” And then once a quarter, the system just runs all the numbers and spits out your risk scores, right? Like totally automated. And the only time you need human input is explicitly when a change has been made and the only human need to give input is the one who made the change, right?

Building that system yourself would require like scanning all your AWS logs. Figuring out what changed. Figuring out like who to message. How to message them? Where to store all this information? Like that's a huge pain. That's like a quarter or a year-long undertaking. Sym has a template for that. So you can very quickly roll out an integration with your infrastructure's code kind of set up and have this whole automatic continuous risk assessment product that will automatically DM on Slack or email or wherever you want. People who are making changes will automatically collect information and risk scores, and once a month or whenever you need it, will spit out that entire spreadsheet without having to kind of do the whole crazy board room meeting, right?

Again, it's like that is the workflow someone would build themselves if they have the time. We're not coming up with something novel here. What's novel is that you can roll it out in a matter of minutes instead of having to spend months or quarters building it. And you can do that without sacrificing kind of bending that workflow exactly to what you want it to be. So does that kind of help make it kind of really concrete of another example of kind of a workflow here?

[00:29:44] JM: Yeah, definitely. How do you see your company evolving over time?

[00:29:50] YM: Yeah, it's a really great question. We’re starting out with some of these like very common security workflows. We're building this kind of like platform. And one piece I haven't mentioned yet is we also are trying to be kind of the bridge between like security and compliance. So I talked earlier about how kind of a problem people face is like around audit time or when every time you have to demonstrate your compliance is mad dash to like collect logs, run queries, do all that crazy stuff super last minute.

When really you kind of want to be baked into the tools you're using. So the cool thing about using Sym templates and Sym workflows is that every time you roll out a new workflow or a new use case, you don't have to write a single line of logging or auditing code. It's all built into the template. So on-demand, whenever you want, you're able to kind of like show log, show reports, show whatever you need to demonstrate your compliance without having to like do this mad dash.

And so you start putting these pieces together, the building blocks, the flexibility, the kind of integrations of different systems and the fact that it automatically can kind of like showcase its own compliance to the world and you start building up this kind of like de facto place for both codifying and demonstrating your compliance. And that's kind of where we want the platform to go, right? We want it to be the go-to. Anytime anyone in an engineering org has like even hears the word security compliance, I want the first thing they think of is like how can Sym help with that? And normally the way Sym can help with that will be, “Hey, we’ve got a template for that.” And also whoever's breathing down your neck to kind of show that you're following the rules, you can just point them to a Sym dashboard, right? So the kind of one-stop shop for anything security or compliance related on the engineering side is kind of the ultimate vision here. But we're starting this kind of very specific workflows.

[00:31:36] JM: Gotcha. Let's talk about actually building Sym yourself. So what have been the hardest engineering problems you've had to tackle so far?

[00:31:46] YM: Yeah. It's been really crazy so far. I think like in all of our last jobs as founders, we were building products that have like very serious like technical difficulties, like really interesting engineering problems. But, ultimately, the user-facing like part of the product that was exposed was pretty simple, right? Despite all the crazy logic from my last company behind-the-scenes of like routing messages across different providers and making sure things are delivered in order and all these kind of like crazy guarantees we get to make. At the end of the day, what the user saw was like a web app where they typed in messages and hit send, right? And they got messages back.

And so the service area for the user was powerful, but simple same, same for my co-founders. n this case, you can imagine there's this crazy kind of engineering challenge just building this generic workflow system that can safely kind of incorporate user code and it's flexible enough to let us really rapidly iterate on templates and integrations. And that's like a whole challenge in and of itself.

But on top of that, I'd say the hardest part has been the surface area and the interface, what we're exposing to the user, is very complicated. We have this Terraform provider with seven or eight different kind of like primitives of different resources you can provision that all kind of string together to have to give you this whole workflow set up. On the integration side, we're supporting integrations with tens and eventually hundreds of external services all with like different APIs and different guarantees and we have to kind of got to have unified interface to all that.

And so almost like the developer experience and the interface design of how we're exposing things to users has I think been the most interesting and tricky technical challenge in addition to the more kind of expected challenges of just like the whole engine we're building behind-the-scenes.

[00:33:31] JM: Interesting can you go a little bit deeper about how you solve these challenges?

[00:33:36] YM: Yeah, it's tricky. I think a large part of it comes from, again, like having a team of founders and engineers that we brought on the team now who have lived and breathe these problems before. So it's a lot of throwing things at the wall and seeing what sticks. Like, “Hey, to provision this workflow, we think here is like kind of like the API and Terraform you'd want. You want to specify these options and you'd want to be generic in this way. Let's say it's like the approval workflow. You probably want to specify the fields that show up in the form when you request things and you probably want to be able to specify the set of resources that you can request access to.”

And then you just like start running through scenarios like, “Does that work for protecting SSH access when someone's using AWS Session Manager?” “Yes, it does.” “Does it work when someone's using Octa to protect access to like third party SSL logins?” “Yes, it does.” “Does it work when someone is using AWS SSO and they have permission sets across multiple AWS accounts?” “Oh, no. It doesn't. We actually need to like specify the like cross account role that gives us access to their AWS SSO instance and we have to specify which account they're requesting access for a permission set,” or all these like very specific things. You kind of just start throwing things at the wall and trying to break your model of what you've designed until you end up with a kind of interface for a given integration or template that actually makes sense and kind of like holds up to the test of many, many different workflows.

And once we establish that internally, we then go through the whole thing all over again externally. So every week we'll have people coming in and we'll say, “Hey, pretend that it's your job to use Sym to do X.” Like, “Here are the docs. Tell us if you can do it.” And so it's very kind of iterative process we're really just like trying things and seeing what works, but it's tricky. Getting that that complex interface right is really hard and it's something that we're kind of looking for people who find those kind of problems exciting, because it's a lot of work.

[00:35:36] JM: The amount of integrations that you have to write to make these workflows easy to create and customize, is that overwhelming? Do you have a scalable way of creating these integrations?

[00:35:49] YM: Yeah. I think we're taking inspiration from a lot of prior art here. Like there's a lot of like technical work that people have done in general. It's like making integrations a little easier to build. So like a lot of like other people have done this we can learn from. There are products that inspire us. I think Rippling, it's a very different domain. They're like HRIT software. But I think their approach to how they let people set up integrations, how they integrate and kind of give a unified interface for things is really heavily inspired us.

And so we're learning from a lot of other people out there in terms of how to think about this. But, ultimately, I think our secret weapon is we've really heavily kind of pre-invested in architecting the system such that it is dead simple to write the marginal integration and to kind of roll it out. For two reasons, one, because we can then kind of scale everything out to a team where you don't need a ton of context or even be a very senior engineer to be able to very quickly add integration to the same platform so we can have a lot of people who can be doing that. And we’ve invested internal tooling and abstractions that enable that.

The other thing is we eventually want to open up the community and have our customers be able to contribute their own integrations. And it's not our customers job to be experts at writing Sym integrations. And so we want to make it as easy as possible for them. That's been another motivation to kind of – It's almost a meta thing where we have to think really hard about the developer experience of writing integrations in order to kind of smooth that process out over time.

[00:37:19] JM: Have there been any integrations that have been particularly difficult to create?

[00:37:23] YM: Ooh, that's a great question. So far it's been pretty smooth sailing. I'd say the one integration that we've struggled a bit with is a lot of our customers – So one of our customers is LaunchDarkly. I'm sure a lot of people are kind of familiar with them. And they use us among other things for protecting access to protecting SSH access to specific instances. And the way we do that with them with a whole host of other customers is via AWS Session Manager, which kind of lets you effectively open a WebSocket to an instance and do an SSH connection without dealing with kind of keys and all the other stuff you normally have to deal with. And that's great.

We kind of have a CLI that wraps that and that kind of serves our SSH integration. The problem is there have been various points where it's become pretty apparent that parts of Session Manager are, call it what it is, kind of a half-baked product where things go wrong in very weird ways. So we've had a lot of really interesting scenarios where we are deep in the code, trying to like decompile binaries that are shipped by AWS. To try to figure out like what is this code doing wrong that's causing this weird issue that our customers see one percent of the time.

One example of that is like – This is a random story. Like we had this issue where we had a CI server that was running this AWS Session Manager kind of integration test, and we would notice that over time the test started failing more and more, but we weren't changing anything about it. And then we would provision a fresh instance and point it to the fresh instance and it would pass every time, and then over time start failing more and more. It turns out what was happening is there was a bug in this like binary helper that was shipped by AWS which wouldn't close a manager session after you kind of terminated it from the client, which meant that you had these orphan sessions that were just dangling there and over weeks and months they would accumulate and cause resource starvation on the instance. And we realized that was impacting all of our customers.

So we went this whole process of reporting and getting it fixed, but it's like not the kind of thing you'd expect to be dealing with in your building's integrations. So I think we're often hitting the limits of some of these experimental technologies, and that's kind of been a bit of a challenge.

[00:39:35] JM: How closely do you need to work with your customers? I mean your product is somewhat young. Do you have to follow them pretty closely or is it more self-serve at this point?

[00:39:48] YM: So, it's an interesting balance. The entire product is designed to be very self-serve. Eventually, we want to get to the point where anyone can you Sym, kind of self on board onto it even. And so the whole thing, like our docs, we really put a lot of effort into them. The whole thing is meant to be kind of self-documenting and anyone can go in and do it. Having said that, you're right, it's a young product. We're still figuring out some of the abstractions. And so for our benefit, it makes a lot of sense for us to work very closely with some of our customers and basically just observe them. We try to like do custom work for our customers, but instead kind of give them the tools they need and kind of like watch what they do and help guide them. And that really helps educate kind of our set of tools and abstractions and how we can improve that for the next customer. So I'd say it's a very collaborative thing right now not because it necessarily has to be from the customer perspective, but because it really helps educate our product.

[00:40:43] JM: Gotcha. And have you learned anything interesting from talking to customers? Like are there any ways in which you're seeing application development change in real-time?

[00:40:56] YM: Yeah. I think a few things. Let's see. First of all, on the application development side, I think it's like people are thinking about security privacy implications much earlier in the application life cycle now. So like, before, securities are like an afterthought. Like someone else will deal with it. Now you have like engineers all around the company who are coming to us because they like have used Sym for some other workflow at their company and they're saying, “Hey, I'm building this other thing internally and I was just thinking about security and I think we need to protect this or we need to do this. Like is there a Sym workflow I can help with that? Can I integrate Sym into this thing I'm building or into the kind of like governance around the thing I'm building?”

And so it's really cool just to see like actually the average engineer has a lot of cool ideas around security. And if they only had a really easy tool they could reach for, I think it would actually be a lot more practical about the stuff and it would solve a lot of the kind of cultural issues we see in our industry around security. And so that's been really inspiring just to see that we can actually be a resource for people and empower them to have something, some low-hanging fruit they can reach for to kind of like experiment with the kind of like some of their security concerns and needs. That's been one.

The other really cool learning or interesting learning we've seen I think on the more like DevOps infrastructures code side is this is something we talk a lot internally, it's like it's crazy. You have these standardized tools right now. A lot of people using Terraform, using Cloud Formation. You have all these different things, and like we initially made the assumption like standard tools means people are using the tools in a standardized way, but we couldn't have been more wrong. It's just really remarkable to us to see from customer to customer how much it can vary and how vastly differently you can use the same tools.

So the way people are structuring their code bases for their infrastructure as code and their Terraform setups and and how they're kind of provisioning resources and what their software development life cycle looks like around this whole like infrastructure as code thing. It varies so much from company to company and it's been really cool learning for us. It's made onboard people a bit challenging because we can't just give a one size fits all like, “Hey, run this script that integrates with your existing setup,” because everyone does things that are so different. And so the takeaway there from us is we'd had to make even the way you deploy Sym very modular to be able to fit into kind of many different kinds of workflows and setups.

[00:43:16] JM: As we begin to wind down, I'd love to get your perspective on the future. Just generally speaking, you've done a lot of different things in software. You've worked at a variety of companies. Do you have some broad predictions about how software development is going to be changing in the near future, and I guess more specifically around security?

[00:43:38] YM: Yeah. I mean I think – So I've got this like thing, this analogy kind of like to give, that I really think reflects the future. And so the comparison I like to make here is if you think about five or ten years ago, a lot of like infrastructure and like deployment stuff was owned by IT, right? Sys admins who were kind of like doing their thing and applications. You're generally thinking about this. So you write the code and like someone else gets into production for you. And this whole kind of like revolution of engineers reclaiming what we now call like DevOps and bringing this into the fold and saying like, “Hey, actually, infrastructure, deployment, like this is something we're going to own now.” But in order to own it, we need a set of tools. And we started building tools, right? We built Chef and we built Docker and we built Kubernetes and we built everything came in between. And so that's really cool. It's really inspiring that we did that.

I think we're seeing that happen now with security. I mean quite literally, like DevSecOps is a thing now, but like more broadly, we're seeing engineers kind of wake up and say, “Hold on a second. Security is not something I can throw over the fence and like someone else to deal with. This is part of my product. This is part of what I do.” And we see engineers kind of reclaiming security.

But just like with infrastructure, in order to do that, people need the tools. They need the things that kind of empower them to actually reclaim and operate in these domains. And so I think what we're going to see is a huge wave of people building security tools for developers to help them with that wave of kind of reclaiming security and hopefully we're the first kind of step of many in that direction.

[00:45:19] JM: Okay. Well, Yasyf, is there anything else you'd like to add? Any other topics you'd like to explore?

[00:45:25] YM: No. This is pretty comprehensive. Thank you so much for all the really thoughtful questions.

[00:45:30] JM: Absolutely. Well, good luck with Sym, and thanks for coming on the show.

[00:45:33] YM: Thanks so much.

[END]