EPISODE 1469

[INTRODUCTION]

[00:00:00] JM: There is increasing regulatory and consumer pressure on companies toa do a better job protecting sensitive customer data. Despite this pressure, data breaches and compliance issues continue to plague the tech industry. Companies like Apple, Netflix and Google have solved these challenges by pioneering a new type of technology, the data privacy vault. Skyflow is making this technology available to all companies through a simple API.

Sean Falconer joins the show to discuss the features of a data privacy vault and the data security and compliancy challenges that it addresses.

[INTERVIEW]

[00:00:35] JM: Sean, welcome to the show.

[00:00:36] SF: Thank you. Great to be here.

[00:00:38] JM: You work on Skyflow. And in order to get into what Skyflow does, we should talk about the constraints of building in regulated industries. If you create an application that is doing something in government, or in healthcare, or in, really, any regulated industry, finance, there are certain constraints around what you can and cannot do. Can you describe broadly what are some of the constraints of working in a regulated industry?

[00:01:18] SF: Sure. If you take, for example, something like healthcare, then at least in the United States, we have a regulation called HIPAA, which is Health Insurance Portability and Accountability Act. And even though that's a United States healthcare compliance law, it also is used as a reference in a lot of areas of the world as well. And it's essentially a federal law of national standards that talk about how, essentially, patient data, patient healthcare data, has to be secured.

And there's a bunch of HIPAA identifiers that are considered PII, personally identifiable information. And if you're operating in that world of health tech, then you have to essentially

comply with de-identifying these various identifiers as well as a bunch of other rules for something like HIPAA.

Similar in fintech, if you're dealing with banking and credit card information, there's a very well-known standard, which is the PCI standard. And there's a lot of different requirements there in terms of how you handle financial data. For PCI, for example, there's different requirements, things like that you have a firewall that exists and is maintained. You don't use vendor default passwords, you encrypt the data, and so on.

The more regulated these areas of business that you're operating in, there's sort of more requirements put on top of operating the data in those industries. And you have to comply with those, or you will receive fines, as well as you might not even be able to operate your business depending on what specifically it is.

[00:02:52] JM: And I would say, historically, the way that those regulatory compliant engineering problems got solved was with ad hoc solutions. And you could get a consulting company to review your infrastructure for compliance, which is obviously an encumbrance. I would say, more recently, there have been infrastructure providers that come with regulation or regulatory compliance built in.

When I go to get some infrastructure that has been made compliant for regulation, what are they doing differently that would not be the case if I was to just go get off-the-shelf databases, or S3 storage, or compute?

[00:03:53] SF: The difference would be that – I mean, it depends a little bit on the specific use case. But let's say something like in fintech. If you're taking sort of off-the-shelf infrastructure that has been deemed to be PCI compliant, then that infrastructure and that company has already gone through a process to buy a third party to actually say that they meet the requirements of PCI. And there's four different levels of PCI, where PCI level one is the sort of highest level. And those are all dependent on how many sort of transactions that you're doing in your business.

The difference would be that that infrastructure has already met the compliance rules and regulations and requirements to meet whatever that particular industry has set as a standard.

PCI is a very common one. A very common situation would be that if you're building an application today, you don't necessarily want to – And you need to take transactions. You don't want to have to store credit card data and deal with that sort of stuff on your own. You might use a company like Stripe. And then Stripe is going to provide you out of the box PCI compliance, because they've already met those different regulations. And you're essentially relying on their infrastructure that has already met those regulations to house and store your credit cards. And they have sort of the domain knowledge and expertise to do that.

[00:05:14] JM: As we said before, you work at Skyflow, and you're building specific solutions for GDPR, HIPAA compliance, SOC 2, PCI. Let's go into one specific system that Skyflow works on, which is the concept of a data privacy vault. Can you explain what a data privacy vault is?

[00:05:38] SF: Yeah. A data privacy vault is a secure isolated database designed to store, and manage, and use sensitive data. In very simplistic terms, you can kind of think of it like a specially designed database that's been designed for storing sensitive data.

And with that, there are a number of requirements beyond just that it's a database. It's built really around this principle of isolation, which is a well-known concept in encryption or API key management. Generally, with encryption, you don't want to store your encryption key with your data. You isolate that from the data. Or when it comes to API keys or, for example, like a database password, we're not checking that into the source code of our application. We're separating that and isolating it somewhere else through something like HashiCorp data vault or maybe AWS Secrets Manager.

This principle of isolation is very common in those industries. It's less common when it comes to, essentially, customer PII. That's one of the sort of principal components of a data privacy vault, is that our sort of philosophy around customer PII, PCI, PHI, is that that data is special and needs to be treated as something special.

And as part of that, it shouldn't be essentially intermixed with all your application data. And we can get into the problems that that causes. But you need to essentially isolate that and protect it differently. And with that comes other requirements. If you're going to isolate it, well, then you need ways of protecting it. You need encryption, tokenization, data masking and other privacy

preserving techniques built in. It has to be very reliable. This is core infrastructure where you're storing your customer data. So it needs high availability and throughput.

You need ways of actually accessing the data. You need things like data governance built in, zero trust architecture. You need to be able to control programmatic access to the data on a table column, row, or maybe even TTL basis. Support a wide range of different use cases, not only adhering to different compliance laws, but solving a wide range of privacy issues like logging PII, collecting PII, data pipelines, credit card storage, usage.

And then the big thing, of course, when it comes to data privacy and security, like, it's great to secure data. But ultimately, the reason we collect all that data is to actually use it in some way. We need ways of using secure data. We need things like privacy preserving analytics, secure cloud functions, polymorphic encryption, which is a technology that we developed at Skyflow that allows you to run operations over encrypted data. All this sort of components or features are like core to the concept of data privacy vault allowing you to essentially create an all-in-one privacy solution, which we deliver through, essentially, a simple API.

[00:08:16] JM: If you talk about keeping the – For example, like the encryption case. I guess the entire database is encrypted at rest? Was that one of the compliance regulations you have to keep for something that needs a privacy vault? You have to keep everything encrypted you have to keep the keys away from the database?

[00:08:43] SF: Certainly, encryption is part of some of the compliance laws. One the requirements for PCI is that the data is protected. But even beyond that, I think that when it comes to data privacy, data privacy isn't purely about – Like, compliance is also kind of the right thing to do for your users. Nobody wants their home address and phone number showing up somewhere in the dark web. And it could be that in some areas of the world, that data is not necessarily regulated by some privacy law. But it's still something that is sensitive to, essentially, your customers.

I think a lot of this stuff goes beyond just simply whether you're complying with the laws of your local government or the industry that you work in. But actually, wholesale sort of protecting your customer data.

[00:09:29] JM: If I think about like the purpose of encryption, or most security systems, a lot of security seems to be about creating a friction that the attacker would not be able to overcome, or at least would be a deterrent to the hacker. But I feel like if you build APIs that are easy to use, in some sense, it could potentially erode that friction. And if the attacker was to just access the secure system through those same APIs, wouldn't that potentially defeat the purpose of a secure system, of a secure API?

[00:10:22] SF: I think you raised a good point. Obviously, that – I think, historically, there's been thought of this tradeoff that where more secure means like harder to use. And I think one of the challenges of developing Skyflow has been making it so that security and privacy can be easy to use. Because one of the challenges that happens when something is hard to use is that people might not use it at all, like from an actual development standpoint. Like, the application developer, if they don't understand it. Then maybe they work around it. Or they make a mistake by accident or misuse in some way. Same with like a solution architect or something like that. I think we've had to try to balance ease of use with, still, a lot of security.

But to answer your question around the API. A lot of this comes down to data governance. The core of the data privacy vault, or this approach, is that, within your application infrastructure, you're never ever going to store any PII. And to access information, you're going to rely on APIs to the data privacy vault. But most of your infrastructure doesn't even need – Even if they have a way to call the APIs, they don't actually need – Most of the time, they don't need the plaintext data. They might need to use the data in some way, but they don't actually need access to the plaintext data.

Let's take a simple example where, let's say, you have marketing automation software and you want to send text messages to some subset of your customers through a service like Twilio. Typically, what you would do is that marketing automation would – Somewhere, the CRM would have access to your customers' phone numbers. And then pass the call – Twilio's APIs is passing the plaintext phone numbers over to Twilio. And then Twilio would send the SMS.

But in reality – First of all, when you introduce the vault, the phone number is no longer stored within your application infrastructure, within the marketing automation. A simple sort of first pass solution would be that the CRM calls directly to the data privacy vault to get the phone number and then retrieves the phone number and then sends it to Twilio. That's better than storing the

data locally. But you're still exposing your CRM and your infrastructure to the phone number. And it doesn't actually need the phone number. It just needs to use the phone number in some capacity. In this case, pass it to Twilio.

A better implementation of the same idea is that instead of having it where the CRM has full access to actually pull the plaintext fit data, all it needs to do is send, essentially, a tokenized form of the data and have the right to call the vault to say, "I want the vault to call Twilio with the de-tokenized version of this data." That way, the plaintext of the data is always staying within the isolated and secure infrastructure of the vault. And then the vault can have essentially access to pass that data to Twilio on your behalf. You're really making it so that you're de-tokenizing or you're exposing the plaintext fit data as late as possible. And even if someone stole the API key to make that API call on your behalf, well, now, all they could do is actually call the API to call Twilio with a token. And if they don't know what the random tokens are, then they can't actually make a call to Twilio even on your behalf with the API key.

[00:13:38] JM: I guess what you're saying is, basically, just because you are adding in a simplicity of an access point, it's not necessarily going to be less secure.

[00:13:49] SF: Yeah. It's based on the principle of zero trust. Essentially, no service or user has access to anything unless you give it explicit access. And then explicit access really comes down to the data governance, policies that you create. You can say let's take another example where, let's say, it's like a customer service portal. And the customer service person needs to see the last four digits of the person's social security number.

Well, in that case, you can create a policy that says that, for this particular role, that this customer service agent can only see the last four digits of the social security number for this specific API key. Then the only thing that they can actually see is like the mass value of the social security number. And the infrastructure never ever gets exposed to the full social security number. And, again, if someone stole that API key, the only thing they could do would be to get the mass values of social security numbers, which don't have exploitable value to that.

It really comes down to not only isolating and protecting, but also, using these privacy preserving techniques to keep the customer data off of your infrastructure and then govern the

rules around what actually has access. You want to have really fine-grained control over what, programmatically, services have access to.

Historically, the way things work, even if you're using something like encryption locally, you encrypt the data, but then a service can either decrypt it and then they essentially have access to everything, or they can't decrypt it. It's kind of like a true fault. Whereas, the more ideal situation is that, based on the business logic or the business requirements of the service at hand, you can localize specifically what that one service has access to based on the rules and the policies that you create. And most services don't need access to all your data. Most of the time, they don't need access to the data at all. They might only need access to a very limited amount.

Going back to the customer support example, that person, the customer support agent, might need only access to the social security numbers. But they probably don't need access to every social security number in your user database. They probably only need access to the customers that they're interacting with at that given time.

There was a data breach that happened at Robinhood a few months ago. And in that case, the customer service agent got socially engineered to giving up their credentials. That happens. But then they ended up leaking something like a million email addresses. And why does that customer service agent have access to a million email addresses at the time of a call that probably don't have a million people in their queue? And they might not even need the person's email address. Like, why do they need to visibly see the email address? By creating these rules and policies, you can lock down access. So that even if someone stole the key, the scope of the breach of the leak would is significantly reduced.

[00:16:41] JM: We've talked at this point about the security of the API and some of the architectural principles, the zero trust principles, around securing that API. But what about the database that is actually behind the API. Can you give more light around the architecture of like what database are you choosing? Or can the user bring their own database and then you just put APIs around it? Or just talk a little bit more about the actual architecture of the data vault.

[00:17:14] SF: It's based on – I mean, a lot of it was built from the ground up. Skyflow, through its first two years of operation, had primarily like an engineering staff of like 40 people working

on this problem, building a lot of stuff from the ground-up. But at the base layer, Postgres database has been heavily modified with building a lot of these privacy controls and technology that we developed on top of it. And these are cloud-based infrastructure. It's something that you would deploy via VPC within your own AWS environment. And we help you set up a lot of that stuff. And that could be a single tenant or multi-tenant environment, depending on what your requirements are.

We automatically do key management around the encryption keys, the service account keys that are created to access the APIs. But you can also bring your own keys and manage that yourself. Everything is API-based. So you can you know create defaults, create the schemas, create connections to third-party services all via APIs. We also provide out of the box studio environment, which is like a web-based UI that allows you to do a lot of that stuff as well for experimenting or creating your first fault. Similar idea to interacting with the database and setting up a database for the first time.

[00:18:28] JM: Is the database meant to be a place to store like the entirety of – Like, is it the source of truth for all customer data? Or do you just – Like, maybe you could walk through a prototypical customer use case. What are they actually storing? Do they need to store everything to be compliant? Or are there just certain rows or certain columns that need to be stored in order to be compliant?

[00:18:54] SF: Yeah, that's a great question. We provide sort of predefined templatized vaults for different use cases. For example, we provide a PCI vault, which defines a schema for all the types of things that you would probably need to store for something in the fintech industry. That'd be like bank accounts, credit card information, transactions, and so forth. We provide a vault out of the box for that, which you could then you know modify as you need. That's a very common example.

A lot of customers come to us because they might want to avoid something like PCI lock-in with a specific vendor. Maybe they're using something like Stripe today. And Stripe gives you PCI compliance out of the box. It's great. But you might get to a point where you don't want to rely solely on Stripe to house your credit card information and do transactions. Maybe you can get better deals with other vendors. Or maybe you just don't want to be locked in to a single vendor.

One way of preventing that lock-in is to, essentially, move your credit card information out of Stripe and bring it into the vault architecture. And then you can use something called Skyflow Connections that allow you to actually call Stripe securely through the vault without any of the credit card data or banking data hitting your backend. And we have a number of different fintech partners, like Stripe, Move, Plaid and so on. We can really help you speed up the process of being PCI compliant in the fintech world.

Now, another common use case, is businesses that want to be, let's say, GDPR compliant. For GDPR, a lot of the things that you'd be storing is your customer data. You can think about that. Like, typically, when we architect a system, especially at the beginning, we might create a user's table in our MySQL database, where we start putting people's names, and email addresses, and phone numbers and so forth.

And the challenge with that is – Let's take a simple example, where maybe you have a web application, and then the web application talks to an API gateway, which talks to a backend. And then there's a database where you're storing customer data. And maybe there's an ETL pipeline that pushes the data down to a data warehouse. And then you have some analytics that run off of that.

Well, if you're, say, collecting someone's phone number, then the phone number gets passed to the API gateway. The API gateway probably has logs. And either by design or by accident, that phone number might get logged there. Then it gets passed to the backend where it, again, might be logged. And then you store it in your database on purpose. And then your ETL pipeline picks it up and it gets copied down to your data warehouse and it gets probably exposed within your analytics.

Well, for something like GDPR, where you need to adhere these different rights that your users have, like the right to be forgotten, the right to access, as well as localization or regionalization laws, or residency laws, then it becomes really, really hard to actually do that in this infrastructure where something like a phone number is copied and replicated throughout the systems. Because, now, that's a fairly simple example. But you take like a modern enterprise business, 20% of enterprise businesses have over a thousand sources in their ETL pipelines. Well, you're talking about a massive amount of data that's sort of being copied and replicated everywhere. It gets hard to even track like what data you're storing and where it's stored.

Something like the right to be forgotten, how do you go through that infrastructure and actually delete it? That becomes a really hard problem to solve. And then if you need to comply with residency regulations, how do you extract that data from your existing infrastructure and move it into a server that's in Europe, or in Brazil, or wherever you're operating? That becomes really, really hard to do. And then how do you even secure it?

With the privacy vault, the way that we help companies solve those problems is, first of all, you're creating a single source of truth to your data in the vault. You don't have the replication issue. And within your application infrastructure, essentially, the customer API PII is being replaced by probably tokenized versions of the data. You still have a representation of the data within your application infrastructure, but it's essentially de-identified data.

And then if you need to regionalize the vaults, well, then you create a vault that houses your European customers that is located in Europe, and one in Brazil, Canada, United States, or wherever you're operating. It becomes a lot easier to answer questions about where is my data being stored while it's being stored in the vault? What data am I storing? Well, just look at the schema that's created for the vault. And in terms of security, while the security and privacy controls are built into the vault, that part is taken care of. It really, really simplifies a lot of the complexity that comes with securing existing systems by architecting a system from this way from the beginning.

[00:23:42] JM: Does the added layer of security in a database lead to any considerable latency or any other penalties?

[00:23:54] SF: Yeah. That's a great question. Obviously, I think one of the things I talked about around the requirements of the vault, this is a core infrastructure. It needs to have high availability, high throughput. But there are, of course, as you add things like encryption, and tokenization, and other privacy and security techniques, technology on top of that, there is going to be more latency than just querying a regular database. That's one reason that you wouldn't want to store all your data in the vault. You really want to keep it to the data that's very sensitive.

But by storing representations of the data within your infrastructure, like tokenized data, there's still a lot of operations that you could perform on the tokenized data that locally you normally would. For example, let's say, in your analytics, you have a report that shows the number of delivered SMS to a phone number.

Well, historically, what would happen is you would actually have that phone number somewhere within your infrastructure and then you're associating like the number of delivery receipts with it and then exposing that within your metrics dashboard. Well, instead of having the problems that come along with having to secure the phone number in this case, you can store a tokenized representation of the phone number. You can still perform the same calculations and operations on the phone number against the tokenized data without exposing your system to the problems of actually trying to secure it.

You're getting privacy security with the vault. It's probably not going to be as quite as performant as a native, well-scaled MongoDB or MySQL database. But it's still very responsive. And that's really our goal. But you don't have to actually retrieve data that often from it for most use cases, because you can rely on sort of de-identified versions of the data.

[00:25:46] JM: Can you give more description for what went into the engineering of the privacy vault? Because when you talk about a 40-person engineering team working to implement these zero trust APIs, I guess, I'd like to know kind of how you architect an engineering team around that. Obviously, there's a lot of work to review, and to test, and to build the key management systems and the other kinds of security around that. But maybe you could give a little bit more sense of the division of labor around building a privacy vault.

[00:26:30] SF: I mean, we have sort of teams broken up into different squads that own different parts of the infrastructure. There's the data plane or database layer. There's the API layer. There's the sort of developer experience that includes the API, but things like our studio that we provide, our governance engine.

Each of the features of the vault essentially become part of the engineering response – Like, the breakdown of the engineering responsibilities. We have a team that's focused on the governance part. We have a team that's focused on the vault architecture making sure that it is scalable, has that enterprise-grade performance, as well as things like our encryption layer.

One of the technologies that we created at Skyflow is a technology called polymorphic encryption, which the name comes from the computer science concept of polymorphism, where, essentially, a single interface represents many data types. In polymorphic encryption, data is encrypted in multiple forms with multiple keys with specific functions for the data. And that allows us to provide operations on encrypted data.

Normally, when we think about encrypted data, we want to encrypt data at rest and during transit. That's pretty standard stuff. But to actually use the data, we have to decrypt it. But with polymorphic encryption, you can actually perform operations like aggregation, comparison against the actual encrypted data. It's another sort of additional layer of security. And, obviously, that takes a lot of domain expertise and incredible amount of engineering to build something like that.

[00:28:08] JM: Polymorphic encryption, to generalize it, if I recall, it's a form of encryption where you're able to prove that you have the key without actually exposing the key or something something like that?

[00:28:22] SF: No. The idea is that – If you've heard of homomorphic encryption, which is kind of has long – Been considered sort of the holy grail for encryption. With homomorphic encryption, essentially, you can perform any kind of operation in like a conventional database over the encrypted data. But the challenge with homomorphic encryption is it's incredibly slow. Like, IBM built a homomorphic encryption database, like proof of concept, I want to say, six or seven years ago. And it takes like three minutes to 50 query records. It's not something that you can actually use.

But with polymorphic encryption, sort of our key insight was that, when it comes to the type of data that we're storing, we're storing very, very specific type of data. You know, social security numbers, phone numbers emails. Essentially, customer PII, PCI, PHI.

Well, that data is essentially like a data structure. Like a social security number or phone number, we call them numbers. But they're not really numbers. You don't do multiplication with a phone number. You don't do sum with a phone number. Because this type of data – Like a phone number where you have an area code and you have – It's basically broken up into three

components. Then we know the types of use cases and operations you typically perform over a phone number. You might want to aggregate everybody that has the same area code. What polymorphic encryption does is it applies different types of encryption to each component of the data structure so that you can actually perform operations on it.

[00:29:54] JM: I'ts basically a form of encryption where you can actually operate over the encrypted data.

[00:29:59] SF: Yes, exactly. It gives you the power of homomorphic encryption without the downside of the runtime problems. And it works in our space because of the specialized nature of the type of data that we're storing.

[00:30:14] JM: Got it. I guess that's to say that if you have your database encrypted with polymorphic encryption, that means that you could basically translate any query, any normal query. You could transpose that query to a query that the database would actually understand in its encrypted form.

[00:30:39] SF: Yeah. I think a good example is if we take a real-life example, where if I go to a bar and I want to buy a drink, then they're going to want to check to see if I'm over 21. And in the real world, I show my driver's license number. And they see my date of birth in order to determine whether I'm over 21. At the same time, I'm also exposing a lot of other PII, which is like my home address, and my biometrics. Like, my height, my eye color. It's not a great system. But we've kind of applied that same in real life scenario to answering those questions in the technology world as well, where we essentially expose a system that needs to tell whether Sean is over 21, to all the data about myself.

And the reality is, is we don't have to build systems like that. Ideally, to answer whether I'm over 21 or not, maybe the system needs to see the month and year that I was born. But using something like polymorphic encryption, we could actually run a query select count from user's table where name equals Sean Falconer, and age above 21. And if that returns one, then I'm over 21. If that returns zero, then I'm not. You can actually use, essentially, something called zero knowledge proof to determine whether I am over 21 or not without ever actually exposing any of my PII.

[00:32:00] JM: I'd like to broaden the conversation a bit. If we talk about the different compliance use cases, like HIPAA, and SOC 2, and PCI, are there notably different constraints on these different compliance requirements? Or are they fairly similar across the board?

[00:32:28] SF: There are differences. I think PCI is probably the most rigorously defined. And it makes sense, because it's been around for a while. It was initiated by the credit card industry to prevent issues like fraud. It's obviously something that's like really sensitive and people don't want their credit cards being falsely charged. It's highly regulated. And they have very specific requirements.

There's now, I believe, over 115 privacy laws for different countries in the world. And those really, I think, vary in terms of how specific they are about what the laws say. And if you take something like GDPR, which is the most well-known sort of PII-related privacy law, there's things like data residency requirements built into that. Well, that's something that doesn't really apply to something like HIPAA or PCI. And that is like a big hurdle for certain companies, especially due to some of the issues I was talking about earlier where, essentially, people's PII is replicated and copied throughout different systems. It becomes really, really hard to sort of disentangle that.

And then there's also these different basic rights under GDPR that your users have, like the right to be forgotten, the right to be notified, the right to be informed. You have to explain what you're doing with the data that you collect. That isn't necessarily true for all of the different regulations.

And then SOC 2 level one and two are really an auditing procedure designed to ensure that service providers securely manage data and protect the interest and privacy of clients. That's kind of a component of some of these other things. But it's not necessarily just because you're a SOC 2 compliant, it doesn't automatically make you GDPR compliant, for example.

[00:34:13] JM: Talking more about the API surface area and the different products that you guys have at Skyflow, we've talked about the privacy vault and the process of securing it. You also have systems for secure Lambda functions, which would mean you could actually have compute logic. What is involved in making – Lambda function, meaning like AWS – As in the AWS Lambda. What is involved in making a Lambda function secure?

[00:34:53] SF: The big thing with the case of like secure cloud functions is that the function is running within the vault environment rather than running within your regular AWS environment where your application infrastructure is. You're bringing the cloud function much closer to the vault and the security controls that are based around that, as well as the built-in data governance rules and policies. All the security and privacy controls that are built in the vault are part of that secured cloud function. And you're insulating, essentially, your infrastructure from, again, touching the PII.

One simple example would be if you wanted to – And I made this example earlier. But we can dig in a little bit further. But, say, you wanted to call Twilio APIs to send an SMS. Well, you can do that securely directly through the vault by calling Skyflow connections with, essentially, tokenized form of the data. And then in our secure cloud function environment, we're going to automatically de-tokenize that data talking to the vault on your behalf, and then talk to Twilio on your behalf, and then pass that data back to your service. But none of the sensitive data, essentially, is ever touching your infrastructure.

[00:36:08] JM: Okay, cool. It's more about the data locality relative to where the function actually runs.

[00:36:14] SF: Yeah. The same sort of privacy and security controls that are placed around how your application can interact with the vault are also placed on top of the secure cloud function. And then it's also isolated and protected within the VPC environment that the vault is housed within, which has tight controls over who and what can access it.

[00:36:36] JM: And you can also have secure data sharing within a Skyflow data vault. If I want to share data with a third-party organization data that lives in my vault, there's obviously constraints around that that come with privacy and security. What's involved if I want to expose some amount of my data in my healthcare application to a third-party? What's involved in making that sharing secure?

[00:37:14] SF: It's the same idea, where – Like, Twilio example, or calling Stripe. But let's take the healthcare example. Let's say that we had to pass some medical records of a customer over to the healthcare provider. Well, then, what's happening is the application infrastructure is

calling Skyflow and the APIs that we provide for making that connection to the healthcare company. This is something that you can configure either through APIs or through our studio environment.

And, essentially, you're making that request. Instead of making the API request directly to the health tech company, you're making the API request to the Skyflow vault passing in the patient data as the identified form of that data to, essentially, tokenize data.

And then we are going to automatically talk to your vault to de-tokenize that data based on how you configure the connection environment. And then we will pass the plaintext values over to the health tech company over a secure connection also based on however their APIs work. And then when we retrieve the data back, if there's PII within that data, we can automatically tokenize it and store it within the vault. So that when we pass the data back to you, you only receive the tokenized versions of the data. Again, it's always about like insulating your environment from being exposed to any of this potentially sensitive data. So that it doesn't accidentally end up in your logs or somewhere else.

And I think like, in essence, the goal with any highly secure system is that essentially tokenizing or de-identifying sensitive data as early as possible. And then you're de-tokenizing it as late as possible. In the ideal world, when you're collecting data, let's say, in a web application, or maybe in a mobile app, that data is never ever touching your backend. You can do that with secure forms that we provide through our SDKs embedded into your website to collect account information. And then that can be sent directly to Skyflow into your vault using the same sort of governance rules that are set up for everything else. And then the tokenized data can go into your infrastructure.

And then when you need to utilize that data to pass it to a third-party, you, again, are never touching it. It's only being de-tokenized, essentially, in transit over a secure connection to the third-party.

[00:39:28] JM: Gotcha. Taking a step back, building applications for financial services or for healthcare has historically been pretty difficult partially because of regulation, but also partly because it's just like interacting with the banking industry or interacting with the healthcare industry is notoriously difficult. But I imagine that, over-time, the gains in speed and ease of use

of infrastructure probably makes interacting with these industries a little bit easier. Do you have any perspective on interacting with regulated industries from a high level? Has it gotten easier over-time or maybe with the changing of the guard maybe in the last years?

[00:40:16] SF: To some degree, I think that one of the reasons you see such an explosion, especially, in the fintech world is, one, I think that there's just like a lot of innovation going on there. Obviously, there's going to be a lot of companies and investment interests there. But they are doing the Lord's work in any way, where they're taking a lot of this complexity around the banking industry, and abstracting it, and providing like very simple APIs on top of that.

To do, I believe, an integration directly with Visa, there's like SDS FTP servers involved. And there's a lot of kind of old-school ways of passing data securely between systems. While companies move and so on, abstract away all that complexity and provide an API layer on top of that. Then if you are someone who wants to build a company that not only does payments, but maybe does some sort of fintech play on top of the banking industry, like a Venmo or something like that, well, now you have these like really simple APIs to integrate with, which is amazing. And I think that's why you see a lot of innovation and also speed to market. The time to market has significantly reduced, because you don't have to go out and build all that stuff from scratch.

It's a similar idea to like the innovation that Twilio really brought, where they took telephony and abstracted away all the complexity of interacting with carriers to do phone calls and send SMS messages and made it a really simple API. And in the spirit of those companies, I think that's also at the heart of what Skyfall is trying to do around data privacy, is make, essentially, data privacy as simple as payments on Stripe or sending an SMS on Twilio.

[00:41:54] JM: Just winding down, can you give some perspective on where Skyflow is going and what systems you might be building in the near future?

[00:42:06] SF: Yeah. We are a Series B startup. We closed our Series B back in November. We're growing very quickly right now. And, certainly, if anybody's listening to this and this is sounding interesting to you, we're, essentially, hiring across all functions, including developer relations, which I am the head of. If you're interested, feel free to check us out.

But I think a lot of our investments from an engineering perspective, of course, is continuing to build up our offering to large enterprises. There are certain things that need to be there, I think, for large enterprises to use a system like this, as well as continuing to improve our developer experience. That's an area that I am very passionate and have a lot of interest in and heavily involved in. We need to make sure that all the components of a great developer experience are there, which includes continuing to improve our studio environment, our APIs, our documentation, our client libraries, and sample applications and demos, and make it as easy as possible for people to really get up and running.

And then we're really continuing to expand the engineering team, which is partly located in India and partly located in the United States around just continuing to build out like our infrastructure and continue to improve performance and add a number of different features from companies.

I think there's a lot that you can do in the analytics world. Polymorphic encryption allows you to perform a lot of privacy-preserving analytics. But there are other things that you can do when it comes to secure data analytics that are areas that we want to invest in in the future. There's a lot of work to be done and a lot of excitement. But I think in terms of where the product is, we have a great offering today for a wide range of companies both in the health tech and fintech space, but beyond that as well, for any company that is privacy-conscious, which should be every company in the world at this point, I think, for technology companies.

[00:44:05] JM: Awesome. Well, thank you so much for coming on the show. It's been a real pleasure talking to you, Sean.

[00:44:09] SF: Yeah, thanks so much, Jeff.

[END]