EPISODE 1384

[INTRODUCTION]

[00:00:00] KP: Consumers are increasingly becoming aware of how detrimental it can be when companies mismanaged data. This demand is fueled regulation, defined standards and applied pressure to companies. Modern enterprises need to consider corporate risk management and regulatory compliance. In this interview, I speak with Terry O'Daniel, Director of Engineering Risk and Compliance at Instacart.

[INTERVIEW]

[00:00:25] KP: Terry, welcome to Software Engineering Daily.

[00:00:29] TO: Thank you very much, Kyle. It's great to be here.

[00:00:30] KP: So to kick off, can you tell me a little bit about how you got your start in the industry?

[00:00:35] TO: Absolutely. I started out a long time ago as a network engineer, and I worked in a lot of IT outsourcing firms and helped with bootstrapping several startups where I wore several hats across the IT ops, corporate security, and network engineering, back in the dark days of the Internet, to help those companies get ready. And I was often the first non-developer technical hire for a lot of small startups in San Francisco in the 90s.

[00:01:06] KP: Well, I've seen technology move pretty quick in my career. There were no Docker containers when I started, for example. Can you comment on any of the major innovations that have really influenced your career?

[00:01:17] TO: Gosh, well, I'm happy to not be doing Novell networking anymore, or working with Token Ring. But you touched on one of the big ones. I think, for me, through the bulk of my career, client server has been the dominant architecture and dealing with the challenges of

Linux, or Mac, or Windows OS. On the client side, as well as various flavors of the server side, it's evolved over time. And certainly, virtual machines changed things.

But for me, the modern push towards bringing the ability to use containers, to use orchestration and instrumentation to really get a picture of the entire enterprise across multiple clouds, public and private, et cetera, as well as on-prem or whatever other flavors you happen to be rolling, that's fascinating to me. The advent of Kubernetes was eye opening for me in particular. It has a different view of how to see the bigger picture of how to govern and apply logic and standards across an organization. As well as being able to really bootstrap using containers, or what have you, to get developers writing code quickly for me is one of the biggest changes I've seen going from taking weeks or days for a new developer to be really productive. To it being days or even hours. That's one of the most fascinating pieces.

And then, from my own perspective, being able to apply standardization and being able to hit the targets you want to hit either from an internal standards perspective, from a compliance perspective, from a security perspective. Being able to do that at scale quickly through centralization, I've found that evolution fascinating. And I continue to watch where it goes, especially as we move towards no-code and low-code solutions.

[00:03:22] KP: So this may be the bias of my own experience. But I often see companies really playing catch up with those sorts of things. They did really well in what you're saying. They made it easy for engineers to code. And now they're victims of their own success going to try and lay some foundations for things like compliance. What do you find is the typical journey for a lot of companies?

[00:03:43] TO: I think you hit the nail on the head. It's challenging as someone who has a deep background in engineering and security to wear the compliance hat. Because we're often brought in nine-tenths of the way through the process and asked to take an existing product, architecture, what have you, and get it compliant, get it ready for someone to get their SOC 2, get it ready to go public so we can pass our SOCs audits, or whatever the business driver is to achieve a compliance objective.

And frankly, that is a recipe for disaster. Trying to bolt on controls, trying to do a checkbox approach to risk assessments, is what I find one of the most interesting and exciting pieces of working in high-growth companies and attempting to do security, and compliance, and risk management. But it's also one of the most frustrating, because if you're not really taking an eyes open view of your risks in the design and discussion phases, if you're not understanding what your CD pipeline looks like, and understanding how the auditors will look at it when they come around to test it. If you're not doing that early, you're really sort of trying to bolt on controls, and they're only going to be as strong as the investment you've made in them.

So I'm not a huge fan of taking a paper mache approach to compliance and hoping it lasts long enough until the auditors leave. And then everyone breathes a sigh of relief and goes back to their real jobs. I'm much more fan of approaching. In fact, whenever I can, I don't even use the word compliance. I try to focus on the risk. I think all of us understand that there are risks we take in software development, in running interesting cutting-edge architecture at scale. There are risks we take, and no one is asking us to not take those risks. But often, it's as simple as documenting those risks, having the discussions with management, and with engineering leadership, and with security, around the fact that we're taking certain risks at certain points in our growth cycle, and understanding that we will mature and we will improve those later. But taking an eyes open view of where are we today? And what makes sense today?

[00:06:01] KP: Well, I realize they've changed the slogan, but I'm curious if you have a response or reaction to Facebook's original move fast and break things philosophy.

[00:06:10] TO: I do admit that over time, certainly before I had to worry too much about security and compliance. And I really only had to do that the first time. I was working at a startup once, and I remember we brought in – I think they closed around the to funding. We brought in some auditors from one of the big four, or maybe at the time it was the big five, to assess our security and our operational practices. And I remember I found it fascinating because they were asking questions that I really hadn't considered. I was certainly much more in the mode of, “Oh, gosh! It's 4am. I've got paged, and I need to fix this problem, I need to get the server back up, I need to get the services running so I can sleep.”

And I am a fan as an engineer of disruptive actions. I do enjoy finding places where you can shave a few seconds off the load page or finding a different route to get to somewhere faster. So what I try to do is I try to merge that with this perspective I have today, which is more focused on governance risk and compliance. And the way I do that is I define the spaces in which we want to play, we want to disrupt and define the places where we do not. I'm sure we all agree that there are good sandboxes and then there are some more foundational elements that we want to be rock solid. And I think that's what I can help to. By defining the spaces that we want to be secure, and have good standards, and be compliant, et cetera, we're also defining the places where we can play.

[00:07:49] KP: Well, to stick with the Facebook analogy, they have the famous like count. And you can see how many times a page has been liked. If you break that, and let's say it doesn't get updated, or it's kind of stuck because some release went bad, I feel like everyone agrees that's not the end of the world. It's not like anyone's going to lose their life or something like that. And it only sort of hurts internally.

As an engineer, I kind of feel like, “Yeah, I will take some risks on move fast and break things if I'm just going to break something like a like count.” But then I'm kind of catching myself and saying, “I don't know if it's my place to make that judgment call.” What's the right organizational structure for risk decision making?

[00:08:27] TO: I think that's fascinating. It comes down to me – For me, it comes down to how we structure businesses in engineering-first companies. And if you think about it, there's quite a linear path through which the business decides to take a certain approach to a problem space. And that is discussed by the business and by product. And then engineering is brought in. And then there's a certain degree of experimentation at that point in terms of how we can solve that problem. What makes sense? And I think sometimes we skip the feedback loop going back to the business of what are the impacts of the decisions we've made once things have arrived in the product hopper and it becomes down to a product and engineering conversation.

So one of the areas that I often find myself in a fascinating pickle is translating between engineering, especially infrastructure engineering, or production engineering, and finance, because often the decisions we make from an engineering perspective have radical impacts for

how the company can recognize revenue, for example, or other business processes that, again, engineering has no idea what finance is doing on their side of the house. And finance certainly has no clue what engineering is doing. But those decisions that are made on each side impact the other.

So I think, frequently, one of the areas I'm most successful is being able to provide a translation service between those two functions. When you make this decision to – Let's go with the example of turning off the light function or something like that. Is there a contract somewhere that the business has committed to delivering that service? What is the downstream impact if we don't deliver that service? How does that affect our financials? What are the finance people have to do to close the books at the end of the month, given that that service delivery was impacted?

[00:10:36] KP: Makes sense. How do those lines of communication typically get built?

[00:10:42] TO: Well, often, they aren't really considered. I think there's an expectation even in fast moving, high-growth companies, startups, et cetera. I think there's an implicit hope that those conversations are happening. But even at the most fast-moving and disruptive unicorn, there's still some silos in which finance is doing their job, and engineering is doing their job.

So one of the ways I often try to solve those problems is bringing people together to discuss a problem in a common language. And I think the common language piece is the challenge. If I bring data engineering to the table to help, let's say, the FynSys people, financial systems and services people, if I try to get them to speak the same language, it can be challenging. But if I approach them with a proposition, that is we jointly need to solve this problem. I think you're able to tap into people's natural problem solving, their natural maybe primates' ability to band together and solve common problems. So it's not particularly sexy. But I do think forming councils, forming committees to solve problems, finding ways to share a view of the world. For example, I recently hosted a forum in which I asked our data engineering team to speak to our finance team and help them build a bridge between their understanding of how they are solving common problems, even though there are so many solving very different facets of that same problem.

[00:12:27] KP: So I know there are a lot of standards out there. I think everyone's heard of HIPAA. Maybe they've heard of SOCs, or I guess it's SOCs 2 or something like that now. What are these collections of acronyms? Like how do they get created? And why do people adopt them?

[00:12:41] TO: It's fascinating to me that there are so many different ways that people come to the table to comply with something. And in some cases, it's a law. It's the price of doing business, for example, with SOCs, which is the Sarbanes-Oxley Act. It's the price of being a public company in the United States in reaction to some of the fraudulent accounting scandals that happened with Enron and WorldCom and companies like that. It was a law passed by Congress. And now if you are a public company, you need to prove that everyone agrees that the way we're generating our revenue and reporting those numbers out to the street so we can sell our stock, they have to be audited by an independent third-party, and you need to comply with the law if you're a public company in the United States.

Another law that applies to certain countries or their citizens is GDPR. If you are doing business with the EU, you need to prove that you've taken reasonable measures to protect the privacy of your users. But sometimes it's less about Congress or a legislative body passing a law. And it's just about the price of doing business. And SOC 2 is an example of that where you are – Let's say I'm providing a service on a startup, and I want to build a better mousetrap. And I want to provide a service to other companies that they can use. But why would those companies trust that I am providing that service securely? That I'm protecting their data? That I'm protecting the data of their customers, et cetera?

Well, rather than just taking my word for it, or doing an extensive first-hand audit of all of our practices, and maybe I don't even want to share all my practices with you because they're proprietary. What I do instead is I say, “Well, look, I can comply with this standard,” the service operating control standard. And I can have a third-party auditor come in and validate that I really am following that standard. And they will publish a report. And then you can read that report for yourself and you have a level of trust that I really am treating your data securely. I'm really protecting the data of your customers. My system are stable. They're secure. I agree to a certain level of being able to continue the business, being able to recover your data in a timely manner,

being able to recover your data at all. So that's another way you can approach these standards is the cost of doing business. And that's where SOC 2 comes in.

And then of course, there's other areas where it's just a barrier to entry for a certain industry. We don't have a law around credit cards per se in the United States. But we do have PCI. And PCI is an industry standard where the big credit card companies came together in response to a slake of fraud that was happening around credit cards and came up with an industry standard. And now if you want to accept credit cards, rather than having to prove to all of your customers and their customers that you're doing a good job securing the data that you're held in trust. Instead, you comply with this PCI standard. And again, you have an external party come and validate that for you.

[00:16:05] KP: So when a company decides to adopt one of these standards, is step one a really long PDF report? Or how do you really get going with the standard?

[00:16:14] TO: Well, one of the fascinating evolving spaces or the evolving pieces in this space for me, is up until a few years ago, the answer was unfortunately, yes. But there are a lot of really interesting startups and high-growth companies these days who are tackling the problem of compliance by building data-driven or automated ways to collect that data. For example, if10 years ago if you said, “How do I comply with SOC 2?” I would say, “Yes. Here you go. Here's a spreadsheet of all the different things you need to do. And then we'll come back around later and validate that you've really done them.” And that would be a very much manual process.

Well, as an engineer, I'm not a huge fan of compliance by spreadsheet. And I'm not a huge fan of manually validating those things with spreadsheets, or what have you. I'd much rather go to the source of truth. I'd much rather have systems pull the data for me and have Python scripts validate the checks, such as if you are – Let's say you have a control around access control. And I want to validate that people's – Let's say a user's account has to be approved to get access to a certain data store before they get access. Well, if the ticket that grants has a date and the access approval has a date, there's no reason I can't throw a Python script at that and say, “Just validate that A happened before B and not afterwards.”

So one of the things I'm fascinated about is the emergence of a whole bunch of tools that promise to make compliance easier by taking a data-driven approach, by pulling that data automatedly. And in some cases, doing the testing for you. I don't think that's the entire picture. If it were, I'd be out of a job. But I do think is a good foundational step to get the core information you need. And then the devils in the details, when the auditors come around and validate, "Alright, well, 24 out of 25 times, it looks like you did what you said you were supposed to do." But this 25th time, the access grant came before the approval. Why is that? Explain that to me."

I think there's still the need to have some human intelligence added to the equation to explain corner cases, to explain break glass scenarios, to help standardize and come up with explanations after the fact. As well as standardization before the fact to provide the best context for how you reach those objectives.

[00:18:52] KP: And in practice, what does that audit looked like? Is it a person coming to the office? Or I guess, today, be virtually talking to developers and going over code? Or is it more like an automated penetration test?

[00:19:06] TO: Oh, interesting. It is often – Some compliance standards do involve a penetration test, for example, PCI to accept credit cards. It's very frequently that you would have a penetration test performed by a third-party, and that would be part of your compliance posture. But the validation step does tend to be fairly manual. It does tend to be sharing evidence with an auditor. And that can be an interesting challenge.

As an engineer, it's interesting to watch auditors perform a code review and walking them through lines of code and explaining a certain feature or function. And how that proves that a certain behavior must happen because it's enforced by code. And then they'll take a screenshot and then they put that in their folder and they're done. So I think this is an area where, on the client side, that is the company side of things, there is the pain felt of complying. And a lot of companies are investing in these automated solutions to make compliance easier, to reach their compliance objectives faster, etc.

And I think the next step is on the audit side of the house. At a certain point, they are going to invest more heavily in finding common platforms through which you can upload the data and it

can be tested directly. Why can't auditors look at our code directly in our GitHub, rather than asking for screenshots, for example? So I think that's the next step in the compliance evolution.

[00:20:37] KP: I see a lot of, especially at startups, companies trying to kind of outsource compliance in a way. You can get a payment processor like Stripe and then you never touch the credit card. Or you can use a certain CRM and say, “Well, we just inherit their security model,” and these sorts of things. Do you have any thoughts on that as a strategy to put the hard parts to a partner?

[00:20:59] TO: I am a fan. I think that there are elements that not just from a compliance perspective, but from a risk perspective. Do you really want to invest in hiring payment engineers and building a large, secure, robust payment system that will weather external attacks and provide all of the data you need to prove to your auditors that it's safe and secure, and prove to your customers as well? Or do you want to outsource that risk? Do you want to pay a Stripe or another vendor and say, “Maybe I'm not scraping every single penny out of this transaction, but I'm willing to pay that cost because I don't have to build a large expensive payments organization, and I don't have to have a good portion of my security team watch it like a hawk. And I'm not responding to all those incidents around payments, et cetera.”

So I think there are a few cases. And frankly, payments is one of them, where it does behoove you to find a way to transfer that risk to another company to mitigate the internal amount of risk. The challenge being of course that no matter how much we transfer the risk to a third-party, it's still going to be – When there's a breach, your company's name is still going to be in the paper on the headline. So I think there's a balancing act between how much can I afford to outsource a business process to a call center or what have you? How much does that make sense in terms of saving me headache, or saving me time, or saving me money, or just letting me focus on the core value proposition that I think our company can provide? Versus the risk of sharing that risk with a third-party?

I remember a few years ago, there was an incident with Quest Diagnostics, which is a medical testing company. And they had outsourced, I believe, all of their billing information to a third-party. Well, that third-party suffered a data breach. And it hit that third-party so hard that the smaller company, that Quest outsourced all their billing to, actually went bankrupt. And I can

only imagine that there was no small amount of distraction and disruption for Quest as it struggled to quickly find a new vendor.

[00:23:17] KP: Yeah, absolutely. There's a lot of issues. I see a common theme in some of these breaches where it's as simple as an S3 bucket was left open.

[00:23:29] TO: That's a huge one.

[00:23:29] KP: Yeah. Do you find that that's the weird corner case they didn't catch? Or is that pretty suggestive of a greater disregard in general?

[00:23:39] TO: Well, I think it's very common in the security industry. Every year, when the Verizon data breach incident report comes out, we call it the DBIR. And this is a fascinating story. Verizon suffered a data breach, gosh, 15 years ago, something like that? And they were one of the first companies to come forward and say, "Yep, we did. We had a breach. And we're going to actually publicize why that breach happened. And we're going to reach out to other partners in the industry and ask them, "Are you having breaches?" And we're going to publish on an annual basis this report of not just theoretical vulnerabilities, but actual breaches that happened over the past year.

So it's a bit like an anniversary for security people. Every time that report is published, every year, we all sort of take a week off and read it, and we pour through, and we understand, "Alright, all this time and effort we're taking to protect the company, is it really going to the right place? Are we protecting against the actual data breaches that happened over the past year?" And I will say it is fascinating to me how much we worry about malicious external hackers who are trying to infiltrate our networks and exfiltrate data and things like that. And that report consistently shows over time that the two major attack vectors are malware through phishing campaigns, and misconfiguration, which is just a fancy way of saying someone didn't set up their S3 bucket correctly. So I think it is very much a case that we are trying to spread the peanut butter a little thinly across all different sorts of security, especially at a startup or a high-growth company. You're likely to have a smaller security team, and they're trying to do a lot. But it is a tough balancing act. Where is the risk of a really impactful breach that could happen? And

where are places where I could really drill in on the likelihood of someone clicking a malicious link in a phishing email? Or how do I just validate that we set up our S3 buckets correctly?

[00:25:47] KP: Well, I think the knee jerk reaction, maybe it's the right reaction, but some quick response to be drastic would be to say, “Alright, lock everybody out. No one, except for Joe, can make S3 buckets anymore. Everyone goes through him. And he's going to set the policy and lock your IAM credentials down really hard so you can just barely read from that bucket.” That might secure the bucket and all future buckets. Is that the right approach?

[00:26:15] TO: That's a great call out. I think there's always a value proposition that has to be weighed carefully. Do I build standards? Do I build enforcement mechanisms that make the speed of development and the speed of releases slower but more secure? Or do I build a culture of trust? Do I build a culture in which my developers are empowered to do the right thing? And in fact, I make it easy for them to do the right thing? But maybe it's tougher to do the wrong thing. It's an option. If I had to break glass if I had to make changes directly into production, because that's the only way to get stable, it's an option, it's available. But it's not the most convenient, and it's not the most culturally supported option.

So I think there is a balance there of what do you lock down in terms of technology? And how much do you build gates into your SDLC that require people to check the box? And how much risk mitigation do you really get from people checking the box, as opposed to building good practices internally so that people have a little bit of peer pressure in terms of what's the value of doing a strong peer review? Is it a checkbox exercise? Or am I really trying to mitigate risk?

[00:27:36] KP: So I know many organizations think of compliance and risk assessment, things like that, is one kind of a growing pain. Not necessarily a positive move forward, but a necessary one. I don't know if that's universally true. But, as a developer it can feel like, “Hey, someone has just come to tie my hands a little bit more than they used to be, hopefully all for the best.” Is there any advice you have for how an organization can mature in an overall productive way?

[00:28:05] TO: Well, going back to something I said earlier, I do think it's important to understand the roadmap of where you are today and where you eventually want to arrive. In a startup, you're going to have a higher appetite for risk. There's less to lose. If you don't have any

customers and you're trying to land that first customer, security is not necessarily going to be top of mind. You're going to be driving as hard as you can to get the business that you need to survive. But once you have that business, you have a duty to protect it.

There's a legal term called due care, which is did I actually perform the duties that a reasonable person would look at and decide that that makes sense? If I chose at a certain point as a very small startup to not have a firewall? Is that okay? Well, maybe. You're taking a lot of risk. But maybe that's the appetite you have as a small startup. If I have 20 customers, and I'm not doing any sort of perimeter defense, I think most people would agree you haven't taken due care. So I think you need to go into an understanding of what risks am I taking today? And what risk can I afford to take? And often that's an opportunity to have the conversation with leadership to have conversation with the business side of the house. What did we sign up to when we got those customers? Did we sign certain agreements that are legally binding? Did we agree to a certain risk that would leave us with a boatload of liability if something went wrong? Or are we taking measured steps to say we are small and scrappy today? Tomorrow, we're going to get a little better, and the day after that, and the day after that."

[00:29:48] KP: Earlier you'd brought up serverless and no-code and some of these new developments in the software world. It seems like that could have a similar story to when we talked about Stripe. That those services could offer me their own to security layer. But I also see it with people wanting to thread so many of these services together, it's just a new set of potential attack vectors. What's it really like working in the serverless, no-code compliance world?

[00:30:13] TO: Well, no-code is fascinating, because that's an area where you are asking – I mentioned earlier, building a culture in which developers are encouraged to do good peer reviews, to good do good code reviews, to check into security about certain approaches. But boy, when you open the floodgates to a whole bunch of non-developers with low-code and no code, a lot of that just doesn't scale. So I think low-code or no-code is a fascinating place to explore for the future, because I'm a huge fan of democratizing the process to give people the ability to ship their products, and play with their ideas and turn them into reality. But that comes with a cost.

If you don't have a level of governance, if you don't have standards, if you don't have a way to validate what settings have been configured, who's allowing permissive access into what sort of data stores? I think that is a challenge for the future that we're just sort of tackling. And I know a few companies that are taking a really interesting approach to security governance with low-code and no-code solutions.

But maybe going back a few years in the past, I do remember when containers and Kubernetes first emerged. There was a real scramble among compliance folks to understand how to even explain to auditors what we were doing technologically and how you could understand what is nginx, for example, and how do we use it? What's the difference between nginx and an OS? How do you prove open source lineage? If I am pulling certain packages, do I know that that project is still being run by the same individuals? Have the original creators abandon it and it's been taken over and perhaps poisoned if I'm still pulling that same product or project into my open source toolkit?

I think that's something that we've started to build some guide rails around. And there are a few tools out there that solve that problem. But other solutions like serverless, and edge computing, and especially low-code and no-code, I think those are areas where I'm happy to say that, often, we are the tip of the spear to have to explain to auditors how these technologies work. And since auditors have to sign off on our reports that these solutions really are implemented securely, or what have you, it's an opportunity to find some creative approaches to securing them to proving that they were built securely. To proving that they can't not be configured securely, which is always a tough one.

So I think that's an emerging space especially with low-code and no-code. I look forward to the next generation of security governance products that will really help ensure that the entire company is secure when everyone's releasing product.

[00:33:21] KP: And when you say prove, is that sort of empirically or there's some clever computer science thing behind it where it's proved-proved? What is proved mean?

[00:33:31] TO: No. I have to say, unfortunately, a lot of it is very low-tech. As I mentioned earlier, there's a lot of low-tech approaches to auditing still. I've worked for some large

enterprises in the Silicon Valley. And even then, we take screenshots with date stamps and things like that to pass audits. So I think there's a lot of room to disrupt here, and I'm looking forward to how the audit firms will approach that as we evolve and grow.

But proof is an interesting one. I think developers and product folks, et cetera, take a positive view of the world. That is we want a certain condition to happen. And auditors tend to come to the table with a stance of we can't take anything for granted. So one of the most interesting conversations I find myself having, especially at startups and high-growth companies, is I will always stand in for the auditors and I will say, “Our auditors are going to ask us a question, like, “If someone wanted to use this tool maliciously, how would you prevent that from happening?” And often the response I get from developers is, “Well, no one will do that. That doesn't make any sense. Why would someone do that?”

So I do see in the evolution of going from the startup, to high-growth, to evolving along that journey of risk, often, we have to ask questions that make developers uncomfortable. What if someone wanted to commit fraud? What if an insider wanted to steal a penny of every transaction? Could they do that? How would you know if they tried to do that?

[00:35:10] KP: Are those standard questions or just good insights you have along the way?

[00:35:13] TO: I think those are scars I have gained over the years. And I just know to ask those questions, because I don't want my engineers who I support to be caught flat footed by that kind of question from auditors. And one of the most challenging things about the cycle of working with auditors is it is a little do or die. And I hate to put developers on the spot. And I hate to ask developers or infrastructure engineers to ask a question or to answer a question that the auditors ask that I haven't prepared them for. So I do try to approach the problem space with a little bit of like a red team mindset that we would say in security, “If I wanted to break this? If I was an insider, and I wanted to use this tool for ill rather than for good, rather than to enable things to go faster? If I wanted to subvert things, could I do that? Would anyone know?” So I think that's just hard won experience. But I do think that's the right approach. If you prepare your partners and your stakeholders, then the audit cycle should be relatively easy.

I think it comes down to understanding what are you trying to do and making sure that you can prove that you set out to do something, and then you actually did it? There's no artificial standard, usually, to which auditors are trying to hold you. It's something as simple as, “Hey, in the early days of the company, we downloaded the security policy off the Internet, and it's boilerplate. But we had to have a security policy. So we just found one, and we did it.”

Well, a year later, the auditors come around and say, “It says here in the security policy that you do XYZ. Are you really doing that? Prove it?” Uh-oh. I think that's the challenge, is that often in that growth cycle, you don't come back around and check that we really are doing what we said we were going to do. And one thing I always advise engineers is don't set the bar too high. If you don't think you can actually meet something, don't put that in your policies. Don't set that as your standard. Because, ultimately, an audit is just a measurement of the distance between what is said and what is done. So if you make your policies so outlandishly high as a bar, and you can never meet that bar, you're going to fail the audit. But if you approach it with some humility, and you say, “Yeah, we don't have the best, perhaps, policies. They're not the most robust. But they are something we can actually do. We can commit to making this happen.” Then you're much likely to have a better time during the audits.

[00:37:56] KP: It wouldn't surprise me to find out that a high-growth company has to do some sort of shoring up. As you said, live up to the policy they established and that sort of thing. Ideally, they've got to get to some sort of steady state or stable point. What do operations look like after that?

[00:38:13] TO: That's really where I have a chance to shine. I love working for high-growth companies. But the first year is usually sort of running around. Just trying to write the ship, build this part out, shore things up, et cetera. But once you can get to a stable state, then I actually get to use my engineering background. And I get to start looking our controls in less of a pass-fail view, because we're so worried about just passing the audits initially. But then we can say, “Alright, well, we passed the audit last year. We're going to have to do it again this year. It was pretty painful. Are there ways that we could make it easier?”

And the two lenses through which I usually look at this is there's the cost to audit. That is how do I actually prove that we did what we said we're going to do? And usually in the first years,

you're just trying to prove that we can do something. But then over time, you should be able to drive that cost to audit down? How do I prove that we did this? Well, do I really need to go and have a whole bunch of interviews between auditors and engineers? Or can I document what our practices are? And then maybe Terry's GRC team can talk to the auditors rather than pulling engineers out of their sprints to talk to auditors. So that's where you can drive down the cost to audit.

But then the other half of the equation is the cost to comply. If I need to prove that – To pick a horrible old-fashioned example, that are passwords have uppercase, and lowercase, and numbers, and letters, and special characters and all that, that kind of old-fashioned stuff. But if I need to prove that for an audit requirement, do I really need to validate that on a manual basis? Or could I just configure our systems to run a script every night? And if any system is not following those standard, I get an alert. Or if I am required to have someone sign off on an access granting ticket prior to provisioning them access, well, couldn't I just put a preventative control in place that says, “Whoa, you can't provision that access, because there's no ticket yet.” So get the ticket signed off. And then you can provision that access. So that's the way, over time, I like to roll my sleeves up a little bit and find interesting ways to pull data directly from logs, set up cron jobs to just dump that data into a data repo, and on a nightly basis, throw some Python scripts against that repo and say, “Did we really do what we said we're going to do? Let's find out. Let's find out ourselves internally before we find out in front of the auditors that we really didn't do what we said we were going to do?”

[00:40:52] KP: Do you have a favorite tool suite of things that you used to get all this stuff done?

[00:40:57] TO: I know some people in my profession are wizards at Excel. I am not a huge fan of compliance by spreadsheets. So I tend to find ways to find the systems that engineers use and take advantage of them. Often, in startups in high-growth environments, that's Jira, or some other ticketing system. If engineers live and breathe Jira, and that's their primary work system that they're working in day-in and day-out, I better have a way to get tasks in front of them using Jira. And I need to be able to have a workflow in Jira that validates for my team that those things are happening on a regular basis. And I need some sort of way to pull that data back and pull evidence back if it's helpful into some sort of store, a data store. And then I am a huge fan of

Python. Just using Python as a quick way, quick and dirty little Python scripts to validate that X event happened before Y event, or A is bigger than B, or what have you.

[00:42:06] KP: And when it comes to these audits, are there any common flags you see most organizations having thrown? What are some maybe quick wins we might be able to take care of and driving that cost down?

[00:42:16] TO: Oh, that's a good one. I think there's really three primary areas that I see as very common challenges. Access controls, getting the right people the right access to do their jobs, is hard enough to begin with. And you're putting a big burden on your IT, or your infra eng, or whoever is provisioning access to various resources. But if you start to move into the space where you're trying to go public, or you need to get your SOX 2, or something like that, the bar raises and you start having to prove not only did Mary need the access that she got last month, but does she still need that access next quarter? What if she moved to a different team? Maybe she doesn't need that access anymore. If she still has that access, I'm going to fail the audit.

So I think access control is a big one. Building some sort of role-based access control or some mechanism by which you have default, allow lists, block lists, to say someone in marketing probably shouldn't have access to the production environment. Finding quick and dirty ways to validate based on the role of the person and the job family they're in what access they really need to do their job. I think investing in that early rather than trying to bolt that on later saves you a lot of heartache.

Another one is change management. Finding ways to take credit for the code reviews that are being performed. Finding ways to validate that people can't comment out failing tests in the interest of shipping something. Finding ways to build your CD pipeline so that it's not just an enabler to get things out at speed. But it also has friction at the right points to validate that you really are following secure coding practices, using SAST or DAST. Finding ways to do some automated testing, and finding ways to lock down the testing suite as much as possible, because there are probably some core functions that I just don't want removed. Maybe I require today, because I'm a small company, every order above $5,000 requires a credit check. Now, two years in the future, I'm going to change that from $5,000 to $50,000. But I'm still going to

require a credit check. So finding ways to lock down those core functions that I don't want someone to subvert is a huge part of an early investment in change management.

And then finally, and unfortunately, this does double duty data recovery controls. The biggest defense you could have against this horrible slate of ransomware attacks that we've been encountering over the past year. Honestly, if you have great data recovery controls, you shouldn't care about ransomware attacks. So investing early in what's sometimes called IT operations controls, and among them is business continuity and data recovery. And being able to prove that even if things went catastrophically wrong, we can still recover our data. And we can still build up from bare metal.

[00:45:29] KP: And how prepared are most organizations to do that? I guess you don't have universal preview, but you've had some sampling.

[00:45:39] TO: I mean, I think it depends a little bit how a startup is formed and the background of the founders. What industry they're working in? I never expected to work in the grocery industry, which is what I do now at Instacart. But I think sometimes different founders who are coming together to solve a problem in a certain space, if they're working in financial systems, they probably are thinking early about are the security of our data, our ability to move data from one environment to the other and things like that. But I find that most of the problems, once you get into the public company readiness, getting your business-enabling certifications like SOC 2, solving industry certification needs like HIPAA, or PCI, every company has to solve those in one way or another. So I think a lot of it comes down to what is the founders and some of the early hires. Have they taken an approach where they're taking big risks because that what makes sense for the company right now? But are you going into that accrual of tech debt in an eyes open fashion, and you're really documenting the things that need to be fixed one day? Not today, not tomorrow, but one day. Or are you just sort of shoving all that tech debt in a closet and saying, “I hope no one ever opens that door.”

[00:47:03] KP: Well, when I think about the common software engineer who probably listens to a podcast like this, I would guess there's someone that had very little training or formal instruction in compliance and risk and things like this. They're also someone who's probably way more excited about learning a new framework than getting some sort of certificate, but it’s

important to your career. To what degree should I, as a growing software engineer, be considering these things? And do you have any advice for how I can stay informed?

[00:47:31] TO: Well, one thing I've really enjoyed seeing as an evolution of the security mindset is my field, governance risk and compliance, is becoming more and more understood to be a core facet of security. You can't just think in terms of red teams, and blue teams, and purple teams, and you need to have a SOC and things like that. You really also need to have an understanding of what are the risks? How do we approach them at different stages in our growth? And how do we build standards that don't bog us down, but provide the guidance that engineers need to do their jobs quickly and efficiently?

So I am delighted to see that security is starting to become more encompassing of GRC. And it's, I think, at the stage now where a startup where you hire your first security hire, there's an expectation that they come to the table with a little bit of understanding of this space, or at least an understanding of what they know and what they need to get some outside help on.

Another place to approach this problem is there are some emerging standards. I'm sure most of your readers are familiar with OWASP. But there's also the CIS, the Center for Information Security, they used to be called the CIS-20, which was a set of core controls that, if you applied them, give you pretty good assurance that you had the right protections for your company, no matter the size. Now, it's called CIS 18 team. They've built it down to just 18 controls.

But going back to the whole idea of due diligence and due care, if you follow a standard like CIS 18, it may be that, for a small startup, you don't need to address every single one of those 18 issues. And you don't need all the different facets of each of those 18 controls. But at least it's a starting place. At least it's a way to say, "Well, we're not going to solve this problem today because it's not staring us in the face. But at least we have a framework through which to view the problem space. And we'll come back around. We'll set a baseline today. And we'll come back in a year, let's say, or every quarter, or what have you, and see how we're doing then. And as we grow, and as we get bigger, and as we have more customers who care more about security, and our data protection practices, and things like that, we have a framework to validate where we were last year, where we are today, and where we need to be in the future.

[00:49:55] KP: Good advice. Terry, is there anywhere listeners can follow you online?

[00:50:00] TO: Sure. You can reach me at terryodaniel.com, and you can certainly follow me on LinkedIn at linkedin.com/terryodaniel.

[00:50:10] KP: Thanks for coming on Software Engineering Daily.

[00:50:13] TO: Thank you, Kyle. This was a delight. Thanks for having me.

[END]