EPISODE 374

[INTRODUCTION]

[0:00:00.7] JM: At Coinbase, security is more important than anything else. Coinbase is a company that allows for storage and exchange of cryptocurrencies. Protecting banking infrastructure is difficult, but in some ways the stakes are higher with Coinbase because Bitcoin is fundamentally unregulated. Isf a hacker were able to siphon all of the money out of Coinbase accounts, Coinbase would have no recourse. This means that it's a much more sensitive problem than the regulated banking system where transactions can often be reversed if somebody hacks in.

Philip Martin is the Director of Security at Coinbase. He joins the show today to explain why his love of complex and high-stakes security challenges brought him to Coinbase. Philip has some specific points around Coinbase and some more abstract points about security that were very useful to me. I really enjoyed this episode. Philip is an interesting personality and is somewhat different than some of the other people I've interviewed on Software Engineering Daily perhaps because of his love of high-stakes security problems, and I think you're really going to like this episode. This is the third and final episode in our series about Coinbase. Our first two episodes covered the currencies of Coinbase and the fraud prevention techniques that the company uses.

We would love to hear your thoughts on this series and any other suggestions or feedback you have. Maybe you have some ideas for other series we should do or just other topics and episodes, or any general feedback. You can send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[0:01:55.3] JM: Spring is a season of growth and change. Have you been thinking you'd be happier at a new job? If you're dreaming about a new job and have been waiting for the right time to make a move, go to hire.com/sedaily today.

Hired makes finding work enjoyable. Hired uses an algorithmic job-matching tool in combination with a talent advocate who will walk you through the process of finding a better job. Maybe you want more flexible hours, or more money, or remote work. Maybe you work at Zillow, or Squarespace, or Postmates, or some of the other top technology companies that are desperately looking for engineers on Hired. You and your skills are in high demand. You listen to a software engineering podcast in your spare time, so you're clearly passionate about technology.

Check out hired.com/sedaily to get a special offer for Software Engineering Daily listeners. A $600 signing bonus from Hired when you find that great job that gives you the respect and the salary that you deserve as a talented engineer. I love Hired because it puts you in charge.

Go to hired.com/sedaily, and thanks to Hired for being a continued long-running sponsor of Software Engineering Daily.

[INTERVIEW]

[0:03:24.6] JM: Philip Martin is the Director of Security at Coinbase. Philip, welcome to Software Engineering Daily.

[0:03:28.7] PM: Thanks for having me.

[0:03:29.9] JM: Let's walk through Coinbase's infrastructure by talking through some basic use cases. When a new user logs on and buys Bitcoin, what happens?

[0:03:41.4] PM: Quite a bit actually. When that new users logs on, of course, they're going through the standard username password check as you with any sight. At that point they do go through two factor authentication depending on how they have it configured. At that point we start to diverge a little bit from what you see to every single site on the internet. We actually go into a system we call device verification that insures that the device connecting to your account was a device you used before. If it's not, we shunt you into an email-based approval workflow. We say, "Hey, this is a new device. Go to your email, click a link." We do that and then it logs

you in from there. At that point you're sitting in your Coinbase account and you walk through the questions by Bitcoin or Ethereum or Litecoin or what have you.

At that point, if you're a brand new user, you're connecting a payment method. This is a credit card, bank account, that gets into a lot of really payment system stuff that I'm really not qualified to talk about.

[0:04:32.2] JM: Luckily, we went to that with Soups.

[0:04:33.5] PM: Oh, outstanding. Those guys have an insane job. Every time I look at the complexity of modern payment systems, the ACH system for instance, it just boggles my mind. Raises my faith in cryptocurrencies.

After you've added a payment method of some sort, at that point you're going to submit a request to say, "I want to buy in number of Bitcoin depending on payment method." It's either instant or delayed essentially on how do we get the money. At that point, we would then acquire Bitcoin on the market at the exchange rate we quoted you when you put the buy-in and we would make a ledger entry that says this person has this amount of Bitcoin. That Bitcoin would then go — Or Ethereum or Litecoin, and would then go into our hot wallet.

We have a two tier system for asset storage, digitalized storage essentially. Hot wallet and a cold wallet. Hot wallet means it is online and programmatically accessible to our internal systems. We automatically send Bitcoin out of it. Bitcoin goes in there initially.

The cold wallet is completely offline. The cold wallet has no programmatic way of moving those coins out of that storage from our systems that require significant human intervention to get those coins out. The nice thing it's asymmetric and depositing coins is super easy. It's a really nice way for us to shunt risk off in this cold storage system. That's really analogous if you want a real-world analogy to the time safes you see in retail stores where tellers or whoever can put money in a slot in top. It's pretty to get assets in, but to remove the assets it's a more complicated procedure. There's a certain timeframe that's necessary. A specific person has access. Things like that.

[0:06:11.7] JM: You mentioned the fact that you're not really involved in the payments side of things. People who are hearing the word security, you're director of security, they might be a little confused. Where does the payments antifraud system and the security begin? Where are the partitions drawn?

[0:06:31.3] PM: That's a great question. It's honestly a little bit fuzzy sometimes. The same techniques that fraudsters may use to defraud us of money are links to techniques that people whose aim is account takeover may use. A lot of the same backend systems we use to mitigate risk of both sides. A great example here is our ID verification process. We use that when you add a payment — Make a payment method to make sure that your name is the same name that's on the payment method and you have to go through and give us a copy to your ID and we do some [inaudible 0:07:01.0] stuff. It's pretty cool.

We use that same system when you want to reset to authenticate on your account if you lose your authenticator device, for example. You'll go through and use and we'll validate your username password. Depending on how your account is setup, validate any other method you may have attached and then we'll say, "Awesome. Give us a scan of your ID and give us a selfie of your face with your ID," and use that to facilitate the account reset process. It's a control that's active on both sides.

In general — You've talked to Soups. There's a whole separate risk organization that's focused on the antifraud use case that we collaborate with extremely closely on the joint space between antifraud and anti-account takeover.

[0:07:47.5] JM: You mentioned this term account takeover. I've talked a little bit to Soups about this. I know that Coinbase sees the leading edge of fraudsters and hackers that are doing stuff and one of this is this account takeover method. Specifically, the cellphone, phone number takeover method. Talk a little bit about this idea of account takeovers and what that actually means.

[0:08:14.7] PM: Yeah, absolutely. It's really, I think, a great public service announcement for anybody who's dealing in the technology space. We see this a lot. A lot of other companies don't see it yet, but it's coming.

Account takeover generically. I guess let's start at the very top and dive in to it. Account takeover is what happens when a third-party hacker is able to gain access to your account. This can be done a number of ways. The most popular way is very very non-technical and very social engineering focused.

What generally happens is these attackers will get a list of people they want to target, they tend to be high net worth individuals. They tend to be people like associated with digital assets in some way. In general, these attackers say, "Okay, that guy is likely to have a lot of money."

Then they go troll public passwords dumps using likely combinations of a person's name to figure out maybe the same email as them as well as depending on exactly what size their target are. They may also troll PII dumps. They get things like your birthday and your mother's maiden name, things like that. Things to help them impersonate you.

At that point they will start pivoting into your online life. Frequently, the first step is a call to your phone provider, your Verizon, your AT&T, whoever where they'll pretend to be you. They'll use some of these PII that they acquired to do that. Most commonly, the last four of your social security number is the default pin on these accounts.

They'll call in. They'll say, "Hey, this is my phone number. I'm Bob Smith. Here is my last four. I want to move my phone from AT&T to Sprint." AT&T is actually, by law, obligated to honor that request. There were a bunch of laws around phone number portability passed in the 90's or so. They have essentially no choice to the matter once the request comes in. They'll go ahead and move it to Spring, to the account number that the attacker specifies.

The first you'll know of this attack is a couple of days after that request has made when the port completes and your phone is no longer on the network of your provider. That's the first indication most people have of this attack happening. Suddenly, your phone can no longer make calls, send texts, receive texts, anything else.

[0:10:24.8] JM: Chilling.

[0:10:25.0] PM: Yes. At that point, the attacker — Normally, that's most what the attacker needs right there. A lot of account reset processes for other sites, like a great example here is Twitter, rely basically solely on phone number to authenticate account resets. You go in and you say, "Hey, this is email." It says, "Okay, I'm going to send you a text." Text, change password. Good to go. The attacker is now in those account and they lever their way towards email access. That tends to be the center of most user's lives both in terms of information that the attacker can get access to as well as hoax into other services.

Once the attacker gains that, methods vary. It could be password reuse. It could be — Another common thing is people's recovery email address on their primary email address. It's normally one that they haven't even considered in years. They haven't thought about it. They haven't secured it. Alumni addresses are very common to have addresses in this case. Just ask yourself, "If the lynchpin of the security of my Gmail account," which has great security. Google does an amazing job at giving users security options, "is this alumni address that doesn't even offer 2FA. What does that say about my risk model layer?" The attacker just needs to breach that. Completely bypass most of the controls on this.

This is in general with the attackers who are exploiting here. This asymmetry of protection where people — This is true across the board. I think this is just true of software engineers as it is of my mom, or my dad. Don't generally sit down and create a threat model of their lives. You don't sit down and say, "Okay —"

[0:12:04.0] JM: I do it compulsively and I still don't do it well. I can audit myself and be like, "I'm doing a terrible job."

[0:12:10.3] PM: Yup. Even if you do it, it's really really had to do and do consistently and do right and do all the time and make the good security decision, not the convenient performance or access or whatever it is.

[0:12:25.5] JM: A large part of that is that social engineering attacks are so hard to model and defend against.

[0:12:32.0] PM: Absolutely. 100%. I think in these cases in particular, it's hard because it's also relatively new. When we think about the rise, the mass rise of 2FA started with SMS. Obviously, 2FAs has been around quite a while, but the broadening appeal, and now there are lots of articles, and every window you need to have 2 factor authentication. What do you do? Well, you can download an authenticator app and scan a QR code and remember to have this thing with you all the time. You can buy YubiKey, but then you have to figure out how to do that and it doesn't work great with your iPhone, or you can get an SMS which is really convenient. People get them all the time.

The asymmetry there to me that's really interesting is, suddenly, SMS, the service that was never designed to be a secure transport for messages is now has to be. It was just never designed to fit that use case. If you were sitting down in the original design meeting for SMS, I doubt if anyone ever said the word message security, really, in this context.

It's an interesting pivot into software security design around — This is not an uncommon case. This happens all the time. You sit down at the beginning of a software project with somebody and say, "Okay, what part are you doing here?" "We're doing this." "Okay." "Here's the threat model. Here are the counter measures. Here is how we think about this." "Awesome." A year later you come back and they've completely pivoted their way over here and the threat model no longer applies. It's really interesting.

[0:13:59.9] JM: You're the director of security at Coinbase which is a company that puts security before, basically, everything else. What are some ways that security permeates the organization in ways that a less security-focused organization might not be permeated?

[0:14:16.5] PM: I'll give you a fun example. A peer company of ours in this space, ShapeShift, experienced a breach and they're very public about it and they had a great instant response report right up. One of the key vectors in that breach was that the attacker, in this case, an insider, was able to gain access to other employees' computers, plan backdoors, steal SSH keys, that kind of thing. When that came out, we're like, "That's really interesting. How do we disincentive that kind of behavior here?"

We built a small Slack plugin, we called pawn board. The problem here is, okay, say you walk up to your buddies, unlocked a computer at work. You want to do something to remind him to be secure, because we all want to be secure. You could change his background, or whatever, but that doesn't really do it for you. Instead, you open up a Slack, you type a quick slash command, /pawned, and then the name and your username. He gets a nice message on a screen saying he's been pawned, and there's a leaderboard where we keep track of this globally. You can see who has been pawned, who's doing the most pawning. We're doing awards thing at the end of every month, like who's gotten the most pawns across the board to encourage this kind of awareness around the risk on locked computers. It's been smashingly successful. The infinite of unlocked computers has gone down drastically. We had multiple people say that they're now super paranoid about do they lock their computer? They're not sure. They're going to walk back and they're going to check, because they don't want to lose the game.

[0:15:42.9] JM: Yeah.

[0:15:43.6] PM: A lot of companies, I think that wouldn't be taken nearly as seriously as it's taken here. When we rolled it out, we did a whole push around why we're doing it and the ShapeShift breach and the underlying point we're trying to make that I think was very very well received.

[0:15:58.3] JM: From an engineering perspective, let's say I'm an engineer and I'm going to push my code to the codebase on Coinbase. Are there additional security precautions where it's like I have to get — Because I've worked at companies where it's like, "Okay. You got to get sign off from two other people on your team of where the code makes it to production." Is there also a security engineer that has to take a look at it for Coinbase?

[0:16:22.4] PM: Not all the time. We hook into the software development lifecycle and a few different places. One is upfront in the designer view. If someone from security participates and every designer view that goes on, sits down with the team in question, walks through a threat modeling exercise and sort of codifies like, "Hey, we're going to ask you to make these changes to the design and here's why and here's sort of how we think about the overall threat ecosystem of this service." "Cool."

The engineers go away, they write code. Every time anyone does a PR within Coinbase, we GitHub Enterprise internally. There's a requirement for peer sign off. This requirement is escalated in line with a sensitivity of the service. If you want to merge a PR on random service, you're going to need a plus one from somebody. This is actually enforced by our underlying merge systems.

If you go and say, "I propose this PR." What we generally do is we'll toss in the team channel on Slack and say, "Hey, I got a PR up. I got to get a plus one." Someone else on your team will go, will do the code review, will plus one. We actually authenticate our plus ones with Duo. They actually type plus on or they type comments under the PR. They then are going to do a push on their phone that says, "Hey, you just plus one this repo, this commit. Is that actually you?" We say, "Yes." Then depending on how sensitive the repo is, they might need two or three. At that point, they can merge that PR.

What also happens before they can merge the PR, obviously we run through CI testing. We also run through automated security scanning on every single PR and we'll block merges on PRs that fail security scanning. This is mostly Stack analysis. Across Node and Ruby, and our frontend JavaScript, but it's also packaged dependency analysis. What are the status of the depths in this PR? Is this thing up-to-date? We also just released internally a packaged credibility scoring system where we say, "Okay, hey, you've included this package. There aren't any CVs. There are no known involvements in this. The last commit was 18 months ago. It has a dozen open issues in GitHub. It's not necessarily a credible package for you to include." We're not going to block deploy because of this. But it'll show up as a warning. But we say, "Hey, you might want to rethink this package because this is like high likelihood for either some sort of package takeover attack or they can then target users to that package or being so and maintain that security sort of creep in the interview overtime.

After all that goes through, passes stack analysis, it passes the DiaMet checks, it passes our grab-based checks of like, "Hey, if you're doing a React project, I shouldn't see dangerously set in our HTML in your code, or if I do, you should explicitly flag it and we should know about it and we should talk to you about what that means. Passes all that, gets a nice green circle in GitHub and you can merge the commit in. That happens to every single PR.

As the service is nearing completion, we'll reengage and depending on the criticality and where the service is going and what kind of data and things it's touching, we'll reengage and do the review of the threat model and make sure that — Like we're talking about before, the thing that they built is actually the thing that they were going to build originally and the threat model still applies and all the same controls are put in place, and they're put in place in a way we agree with and then potentially do a deep-dive pin test on that service depending on the criticality of where it's going.

[SPONSOR MESSAGE]

[0:20:01.6] JM: At Software Engineering Daily, we need to keep our metrics reliable. If a botnet started listening to all of our episodes and we had nothing to stop it, our statistics would be corrupted. We would have no way to know whether a listen came from a bot, or from a real user. That's why we use Encapsula to stop attackers and improve performance.

When a listener makes a request to play an episode of Software Engineering Daily, Encapsula checks that request before it reaches our servers and filters bot traffic preventing it from ever reaching us. Botnets and DDoS are not just a threat to podcasts. They can impact your application too. Encapsula can protect your API servers and your microservices from responding to unwanted requests.

To try Encapsula for yourself, got to encapsula.com/sedaily and get a month of Encapsula for free. Encapsula's API gives you control over the security and performance of your application. Whether you have a complex microservices architecture, or a WordPress site, like Software Engineering Daily.

Encapsula has a global network of over 30 data centers that optimize routing and cacher content. The same network of data centers that is filtering your content for attackers is operating as a CDN and speeding up your application.

To try Encapsula today, go to encapsula.com/sedaily and check it out. Thanks again Encapsula.

[INTERVIEW CONTINUED]

[0:21:45.6] JM: Now, I know you weren't at Coinbase since the beginning, but it's not long since Coinbase has been a brand-new startup. Brand-new startups don't have that level of security that you just described. Do you have any idea what was the roadmap to getting to that level of security? There must have been a lot of initiative on the part of the upper management to get those kinds of programs in place.

[0:22:16.7] PM: I think what really sets us apart in that sense is sloppy cryptocurrency companies don't survive.

[0:22:22.2] JM: They don't sure don't.

[0:22:23.1] PM: You can see this, a guy named Ryan McGeehan keeps a thing called the Currency Graveyard. It's a GitHub thing of his where he keeps a record and it goes into reasons why of every single cryptocurrency company that's had a breach. It's a great project. Very educational. Sloppy cryptocurrency companies die.

I think what you're seeing is the pressure of any small company, suddenly in their cryptocurrency space you're holding a million dollars, $10 million, $100 million. You see the attackers coming at you time and time again and you either sort of rise to that occasion, like Coinbase did and like the early engineers did, or you fall. It's a crucible end.

[0:23:04.7] JM: The upper management realized that early on, I guess.

[0:23:07.6] PM: Oh, yes. Absolutely. It's been central to Coinbase as Ethos since the beginning. Our number one engineering priority, literally, we stack right them. Number one; don't get hacked.

[0:23:18.8] JM: I meant to asked you this earlier when we're talking about social engineering, because this is such a difficult area to defend against. Do you feel like — This is a question more about the broader software engineering landscape. Do you feel like social engineering attacks are increasing in sophistication faster than we can defend against them?

[0:23:38.8] PM: I don't know that they're increasing in sophistication. I think they're changing in targets. They're going after different fundamental pieces of infrastructure, but the same fundamental attacker that's calling into Verizon and pre-texting that they're you and they have this information is the same technique that software engineers have been using since the dawn of time to pre-text into customer support, to get information, or what have you.

I think the big difference that we see is the potential payday for attackers continuous to go up either in terms of raw value or in terms of the value of the PII stole or the value of whatever they're after. As more and more of our lives are available online it's much easier for the attacker to convert that stuff to money. Whether it'd be PII or credit cards or Bitcoin.

[0:24:29.2] JM: PII being personally identifiable information.

[0:24:30.9] PM: Absolutely. I think that is a huge drawer to the style of the attacker.

[0:24:38.0] JM: I read an article about some of the principles of security at Coinbase. I guess we don't want to talk about some of those. One security principle that you pay a lot of attention to is layered security. Explain what layered security is and describe some of the layers at Coinbase.

[0:24:55.9] PM: Yeah. I like to explain this in the context of a Mike Tyson quote. The Mike Tyson quote is, "Everyone has a plan till they get punched in the face." Layered security means start planning after you get punched in the face. Assume that you're going to lose the first battle and build in the redundancies and the controls and the layers of security that allow you to lose that first battle and still win the war.

I also like to talk about this in terms of that many companies where it's profitable to throw a zero day at them, at least like a major zero day. A zero day being a brand-new exploit that's never been used anywhere before. When you think about it, most places you see zero-days or you hear about them used are actually by either governments or a company is closely in lined with the government.

[0:25:44.2] JM: You're saying that because the cost of the zero day is 500 grand at the black market to get —

[0:25:48.1] PM: Right? Government pays for it and they target somebody that's of immense value to them.

[0:25:52.6] JM: Your return better be more than $500,000 or a million dollars or whatever the cost of the zero day is.

[0:25:57.7] PM: Exactly. I think Coinbase is one of the few places that store liquid value, where cryptocurrency is liquid value in a way that can't then likely be clawed back like a traditional financial system. Where it starts to become worth it to spend a zero day in the attempt to breach Coinbase. When you start from assuming zero day, that's where we go back to the Mike Tyson quote, you're going to get punched in the face. The plan has to start from there. It can't start from this nice steady state of, "Oh, we got an IDS alert about a suspicious actor," and blah-blah-blah-blah-blah. It has to start from the actor somehow landed on a frontend box. What do you do now? So that when you do that, you start to build in where a microservices architecture, which is pretty handy in this case, but you start to build in internal trust boundaries around, "Okay, this is our frontend box. This can talk to these other things. Those things are authenticated in this way. Requests like you build in channels for the second tier systems to confirm requests with other first tier systems."

[0:27:04.9] JM: Go in to that in more detail; first tier, second tier.

[0:27:07.7] PM: An example here.

[0:27:08.3] JM: These are the layers you're talking about. Okay.

[0:27:09.4] PM: Yeah. Exactly. In terms of like the first layer would be the service that's talking directly to customers. The thing that you visit when you visit coinbase.com. Internal layers might be the service that handles internal accounting or Bitcoin wallet managements or risky transaction analysis or whatever. All services that that frontend service needs to talk to but something that needs to talk to customers directly. Probably not the place that attacker lands.

An attacker will probably land on a frontend system. Again, if we assume zero day in that way. Ignoring all of the — All their potential avenues for compromise.

At that point, what we want it is backend systems that don't fully trust that frontend system. We want to be able to cross confirm requests and make sure, "Did a user really do this, or is this an attacker on the system?" We want to be able to rapidly detect that attacker. Which, again, microservices are great doing here because they have a limited functionality. They only do so many things. Traditional general purpose server environment. You might have a very active system with lots of different commands and processes and systems running at the same time. Inside of a Docker container or inside of a microservice, you just don't have that. The list of normal things is extremely small, so it can build in alerting that's extremely targeted to anything outside the norm for that service. Then we can build controls on top of that around rapidly responding and effectively isolating and both being able to rapidly recover and effectively investigate anomalies that we see inside the environment.

[0:28:49.7] JM: A lot of the shows that we've done around microservices, we're not typically talking about security. We typically assume a secure system. I'm not sure why we do that. Typically, about preventing outages and stuff, but when we talk about security — By the way, in those shows, a lot of what we talk about is observability. When you're able to — That's the buzzword of the day. When you want observability because you want to be able to turn those your raw logs into monitoring details that you can easily look at in a dashboard and anybody can glance at the dashboard and say, "Here's the health of my different services. Okay, here's the request path. Everything looks good." Does that play into security as well? How does observability fit in with security?

[0:29:34.9] PM: Absolutely. It's absolutely critical. I do want to talk about internally as telemetry, but observability works too. In two cases, number one; in a microservices architecture, you tend to cycle these boxes more than you would in a traditional sort of pets not cattle infrastructure. In that sense, that's both good and bad. It forces attackers to move around, but it also means if as the data and research has shown us that the meantime detection is in the months' range. But the time you detect the actor in your environment, their initial entry of vector is probably been redeployed a dozen times. How do you walk that back effectively? How do you answer questions like, "Where else was the attacker?" if all that data is gone? In that sense, great

logging, great observability, great telemetry I think is absolutely core to security in a microservices architecture.

Secondarily, I think the thing that really helps security microservices is simplicity, is the focus that's implied by a micro-service. A micro-service isn't doing 10 things. It isn't necessarily talking to a thousand different services, internal and external, but has a much more simplified set of behaviors and actions. It's also, from my perspective, better because the code is generally much easier to audit. You talk about a four million line monster of an application, not even the developers know what's going on in that application much less a security guy trying to audit it. Talk about a 200 line micro-service. It gets to be interestingly easy to audit.

The hard part then becomes this system complexity scales immensely. The problem moves from how do I audit this three million lines of code to how do I audit these 3,000 micro-services communicating with each other and trace data paths. How do you do build a data flow diagram in a massive micro-service environment? It becomes very interesting and challenging. It's hard to address that problem, because if you're a threat model enthusiast, the first real stop of any good threat model is that data flow diagram; what are my inputs? What are my outputs, and how is it moving between those two things?

[0:31:47.0] JM: Okay. We're talking about building infrastructure and maintaining it and observing it. What Coinbase does for a lot of users is store cryptocurrency, cryptocurrency is electronic money, so you're storing it in a database somewhere. What is that database look like and what do you have to do around that database? We are kind of talking about the services that are on top of that. What is the actual storage model — I know there's a like distributed cold storage and stuff like that. You could just talk about the actual storage model.

[0:32:24.5] PM: There are two pieces of critical — There's a lot more than that. When you talk about actual cryptocurrency, there's two pieces of information that's critical. One is the ledger, the balances, who has what money. The other are the keys themselves. How do we safely interact both with the hot and cold keys that we use to actually store the currency or, in our case, actually sign the transactions?

The accounts ledger is relatively straightforward. It's essentially a really big database table. We have lots of controls around who can access that and how they can access that and from they're going to access that. We have independent auditing on the access that occur in that and we make sure that like, "Oh, hey. This bounce changed from here to here." There's a separate service looking at that saying, "Does that make sense? Did we see the request come in separately? Is this all closure?"

The much more complex undertaking is around e-storage, especially with the hot wallet. The hot wallet, we have N-number of million keys at every address that's ever been used to receive Bitcoin into Coinbase. We have the key for that address in hot wallet. The challenge is we have to be able to programmatically interact with these keys on a regular basis to facilitate transactions in and out of Coinbase.

At the same time, we don't want an attacker to be able to effectively and easily interact with that if they landed in our ecosystem. It's a challenge that not many other places need to tackle. I think key management on this scale is a pretty rare thing to try to do and it's a really interesting challenge.

[0:34:00.9] JM: Okay. Can you talk more about what that's been like evaluating that challenge and getting more sophisticated at it?

[0:34:08.9] PM: Sure. I think like any other big engineering challenge, it's an incremental approach. Coinbase started Whatever, four or five years ago now and I'm sure the hot wallet design then was nothing like it is today. We think about this challenge in terms of, again, that perspective that an attacker is in this system. The attacker is sitting on one of my server somewhere interacting with our service. Trying to learn more about it. Trying to make it do things that I don't want them to do. How do we make that hard for them and how do we force them to do it in a place that we can see it?

A fundamental aspect of our approach to key security is that an attacker shouldn't be able to extract those keys. In order to interact with our keys, an attacker should have to stay in our infrastructure the entire time. It gives me more change to detect him. It means that he can't just

take and run and send transactions later. It's a pretty important core design piece there, segment out key storage into a service we call Nox. It's unimaginatively named, I know.

Nox's single and only job is to manage keys inside transactions. As you send the Bitcoin transaction at coinbase.com and you type in the address and whatnot, that percolates to this system. It makes changes in the accounting table. It talks to our Bitcoin wallet service. It generates a transaction, and then that goes to this very isolated system, Nox, to actually be signed. Until that point, it's all so much just numbers and logs. Those doesn't actually matter. That transaction isn't good for anything.

As it comes out of Nox, that's money at that point. As soon as exists Nox. We isolate Nox as much as possible. It is, I think — Last I checked is literally something like 300 lines of code. Extremely easy to audit, very straight forward and simple. Very locked down in terms of who has access to it. Who can interact with it. I guess this is an interesting one, intention only fragile in a way that — If you're an attacker, you breach a system. You're looking around. You say, "Okay. What I want to do is I want to do thing X." You're going to try to figure out how to interact with a service that does thing X. While you're doing it, you're going to make some mistakes.

You're going to get a parameter wrong. You're going to make some things out of order. You're going to send a request to the wrong API endpoint, as you explained, functionality. Nox has written to issue 500s at the drop of a hat and throw pager duties when that happens. Intentionally fragile. If you do it even a little bit — Because a human shouldn't be talking to Nox. The program is talking to Nox. Programs are not inconsistent. Programs do it the same way every time in general.

If something is wrong, it's probably because either someone messed up their code and they're interacting with Nox currently, or a human is trying to interact with Nox and is making understandable mistakes because they don't know. They don't know the system the way we do.

[0:37:00.4] JM: Interesting.

[SPONSOR MESSAGE]

[0:37:09.7] JM: Hosting this podcast is my full-time job, but I love to build software. I'm constantly writing down ideas for products; the user experience designs, the software architecture, and even the pricing models. Of course, someone needs to write the actual code for these products that I think about. For building and scaling my software products I use Toptal.

Toptal is the best place to find reasonably priced, extremely talented software engineers to build your projects from scratch or to skill your workforce. Get started today and receive a free pair of Apple AirPods after your first 20 hours of work by signing up at toptal.com/sedaily. There is none of the frustration of interviewing freelancers who aren't suitable for the job. 90% of Toptal clients hire the first expert that they're introduced to. The average time to getting matched with the top developer is less than 24 hours, so you can get your project started quickly. Go to toptal.com/sedaily to start working with an engineer who can build your project or who can add to the workforce of the company that you work at.

I've used Toptal to build three separate engineering projects and I genuinely love the product. Also, as a special offer to Software Engineering Daily listeners, Toptal will be sending out a pair of Apple AirPods to everyone who signs up through our link and engages in a minimum of 20 work hours. To check it out and start putting your ideas into action, go to toptal.com/sedaily.

If you're an engineer looking for freelance work, I also recommend Toptal. All of the developers I've worked with have been very happy with the platform. Thanks to Toptal for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:39:13.5] JM: Okay. Zooming out you have worked on some pretty high-sensitivity companies, Palantir, Amazon. You did counter-intelligence for the Army and now you work at Coinbase. What are the commonalities?

[0:39:30.0] PM: Motivated attackers. I really enjoy being at an organization that has something to protect that attackers also want to get at and are willing to expend the efforts to get it. I think it's just a much more engaging place to be for a security person.

Some of the commonalities, I think, they're all —

[0:39:48.9] JM: You like that because of the stakes or because it just gets your blood going in the morning?

[0:39:55.8] PM: I like it because of the challenge. I guess analogies are great. The analogy is to may be a more traditional software development context. Do you want to write a service to do the same thing that 10 of those services have done before you that's slightly tweaked in a different way? No. You want to tackle a challenge that's never been done. You want to tackle something that has no or very limited hierarchy because it's exciting. It's interesting. You get to break new ground. You get to see new things. Gain new experiences.

I think it's very very similar for me in security. The motivating thing for me is that there's someone else on the other side of this equation in security. There's another human out there somewhere that is trying to do something that I don't want them to do. That human is motivated and they're creative. They're trying new things. They're evolving as I throw counter measures at them. They don't go away and go home and cry. They'd say, "Okay. That sucks. Let's try something new."

The only way to model adversaries in my opinion is as just as smart and as motivated as you are, if not more. That, to me, is really cool.

[0:41:05.8] JM: Is it a game? Does it feel like a game?

[0:41:08.0] PM: Absolutely not. When you say game, I get the context of like the outcome maybe doesn't matter but it's like less important. You played a good game. Congratulations. Whatever. I take it much more seriously than that.

[0:41:22.2] JM: Got it. Interesting. Was there anything unique about the army? Because that's very interesting. I haven't done any interviews with people who are in military. What was unique about that?

[0:41:36.9] PM: I think that's actually where I originally got that sense of —

[0:41:40.7] JM: The seriousness.

[0:41:41.6] PM: Of mission. That desire to be some place where I perceive the thing I'm doing really really matters. It matters for the company. It matters in terms of the significant outcomes. You get a lot of that in the military. To find that outside of the military I think has been sort of my search ever since.

[0:42:00.3] JM: Interesting. Yeah. I listen to some podcasts with military people and I've read some books recently about military stuff and — I don't know if this is what you're working on, but communications in the field, for example, it's just like so complicated and it's like totally unlike almost anything in Silicon Valley software engineering.

[0:42:25.9] PM: Yeah. The amount of friction introduced by trying to do something in a war zone, I think, it's hard to overestimate how hard that is. From the simplest thing, obviously, limited electricity, or maybe more apropos inconsistent voltages. Plug your laptop into that and you're going to fry it eventually. Just the amount of thing that would never occur in day-to-day life in the U.S. that occur in that environment makes doing even the most simple jobs extremely, extremely complicated.

[0:43:03.0] JM: That raised the bar pretty high for the companies you'd be willing to work for after that.

[0:43:07.4] PM: Yeah. I don't think they're directly — You can't make a one-to-one comparison of the military to this living world. They're apples and moon rocks. What I took away from it, and everyone takes always something different. What I took away from it was that connection to the outbound, to the mission, that ability to get really engaged in what's going on and if it's an important problem, be able to attack it with the resources it deserves. As supposed to like, "Hey, that's a problem. I guess it kind of matters. Let's try some stuff, but we're not going to go crazy."

[0:43:50.8] JM: We're a photo-sharing app. Who cares?

[0:43:52.5] PM: We'll time box this at two weeks and move on if it's three-quarters finished. Whatever. That's just not motivating to me.

[0:44:00.8] JM: The last interview I did was with Linda and Jordan about just kind of the direction Coinbase is going in, kind of the Coinbase product, and different currencies. We talked some about the fact that there's Coinbase and there's also this exchange. Do you work on the security for the exchange as well?

[0:44:19.5] PM: Yeah.

[0:44:20.3] JM: Okay. Is there anything significantly different than what we've been talking about for the exchange?

[0:44:25.5] PM: No. The backend systems are very very — In fact, in some cases, they're the same systems. There's not a real difference to us across the two. I think there are some security UI differences because our [pertable] users are different. We want to present security to them in different ways. Yeah, there are not core-back in differences.

[0:44:47.0] JM: Do you feel like there're security issues that you have to deal with change as volume scales up, or do you feel like the trading volume or purchase volume, or do you feel like the techniques that you have in place, security-wise, are scalable?

[0:45:01.1] PM: They're very scalable. I think the issues we deal with the number of volumes is around operational management. For example, the percentage of funds, we have hot versus cold. We don't want to introduce transaction delays because we misjudged how much needs to be allotted in a given time as volume scales has become more unpredictable. That's the major thing outpoint too that has changed.

[0:45:25.4] JM: To close off, just a quick discussion of some incidence response. You wrote about the cloud bleed bug. We did a show about cloud bleed, and this was — For those who don't know, this cloud flare, there was a bug where it was spouting data everywhere and people who are curious about that can go back to that episode. You audited kind of what did this effect

to Coinbase. What was that auditing process as an example of incident response? How does that exemplify how you look at things at Cionbase?

[0:45:58.4] PM: This was an interesting one, because it was almost entirely out of control. This was an incident response where we had to work very very closely with cloud flare and figuring out what's the impact, what's the scope. Cloudflare did really an outstanding job working with us in both being very transparent in their public report too about what happened. What the impact was? What the potential for impact was? As well as they had a team running 24/7 around searching the internet and working directly with the major search engines and archival organizations to remove content that was potentially an artifact of cloud bleed and then directly disclosing to us when they found something that may be Coinbase related.

We obviously did our own due diligence in searching and what, honestly, we found was that Cloudflare was doing this in an insane rate. Every time we found something that might have been Cloudflare related by the time we came back, it was purged or gone. They were really hustling on this. I give them a huge amount of credit for that.

This is the really interesting sort of case study, in a complete third-party breach, where we don't have necessarily audit logs from them. We don't have controls. There's no amount of — There's a limited amount of lockdown we can do. As soon as we understood the bug, we were forced to engage with Cloudflare like, "Hey, let's work, search for this stuff." We also proactively rolled every session that might have been impacted in the time window they specified.

The nature of the bug was that the most likely stuff that was damaging that would have leaked were going to be things like cookies or other similar tokens. We looked at that and said, "Okay, let's roll every session. Let's go take a look at API, key usage logs and notify, potentially impact your customers there of like you used your API in this timeframe. It may have been compromised. We suggest you roll it."

We didn't want to — practically, link rolling sessions is easy. You just log out. They could log back in and good to go. Proactively deleting API keys, in some cases, in parallel businesses. We didn't want to get as aggressive with that as with the sessions. This sort of walks back to a risk-adjusted incident response process. We could have gone full-board, invalidated every

single API key, invalidated every session. Forced the full user base password rest, but based on the specific incident in question, we wanted to do the thing that was both safe and least intrusive to our users while giving them the option to ramp up their response if they felt it was justified based on their risk assessment profiles. That's why we rolled sessions slow and back to users. We notified, potentially impacted API customers and we did the public blog post to help get the word out to users.

[0:48:53.4] JM: Yeah. Okay. Yeah, it's such an interesting case study because, simultaneously, it makes you optimistic and pessimistic.

[0:49:01.5] PM: Absolutely.

[0:49:02.6] JM: Pessimistic in the sense that like, "Oh! Oop! A cloud service," and there's no control over what you can do to respond to it as a customer of that service where you had data breach, but optimistic in the sense that the response was so beautiful.

[0:49:16.3] PM: Yeah. They're a great example, although the underlying sort of concept here that the third-party vulnerability management and third-party breach response is I think a huge area of risk across the board and only becoming more so. I think this especially true for startups. The siren call for smaller companies. The siren call of this third-party company where you can just outsource this thing to, "This is important thing, but it's a thing that's not core to your business, and they'll take care of things for you until they don't anymore, until there's a breach, until there's problem."

Small organization, it's hard to spend the resources to know what third-parties you should trust and then what third-parties might not treat your data like you would treat it. That's the fundamental bar hold for our vendors, and we talked to them, is, "I'm going to give you this data. Okay, you're going to do this thing for me. That's awesome." Are you going to treat it like I would treat it if I held it in my own system, or better? Are you going to cooperate, are you going to be logs? Blah-blah-blah. There's a bunch of sort of ancillary questions that go along with that. The fundamental bar I want to see are you going to treat it like I would treat it?

[0:50:21.6] JM: Okay. Good. Build versus buy evaluation. Okay. Cool. Thanks. Thanks, Phil. This has been a great conversation. I really enjoyed it.

[0:50:27.7] PM: Me as well.

[0:50:28.3] JM: Okay. Thank you.

[END OF EPISODE]

[0:50:33.2] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily.

Thanks again to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver this content to the listeners on a regular basis.

[END]