Zero-Shot Learning - Episode 9: Matt Moore from Chainguard

Build from source

Hosts: Nancy Wang, Chief Technology Officer, 1Password
Dev Tagare, Senior Director and Head of Engineering, Gemini Enterprise and Business, Google (personal capacity)

Guest: Matt Moore, Co-Founder and CTO, Chainguard

Nancy Wang: Welcome to Zero-Shot Learning. I am Nancy Wang, and I'm the Chief Technology Officer at 1Password.

Dev Tagare: I'm Dev Tagare. I lead engineering for Gemini Enterprise and Business at Google. I'm here in a personal capacity, and these views are my own.

Nancy Wang: Today we are joined in our hybrid studio by Matt Moore, co-founder and CTO of Chainguard. Matt spent over a decade at Google building foundational infrastructure across the Kubernetes ecosystem, and he co-founded Chainguard in 2021. I want to understand how that all started, including the team behind Sigstore, after seeing firsthand that open source infrastructure tools weren't making it into production at the scale the problem required. Welcome, Matt, to our studio today.

Dev Tagare: Yeah, thanks for having me.

1. From Google to Chainguard

Nancy Wang: You spent over a decade at Google building, you know, foundational infrastructure. We talked about obviously Kubernetes, you know, household word here for us DevOps folks just call us. Right. And after leaving Google you also contributed to the development of Sigstore. So what did you see in all of that work. Right. And where did this idea around, you know, start sort of secure from the very beginning. Right. Shift left come from.

2. Building trust from source

Matt Moore: I think a lot of the ideas behind it. You know, you mentioned, you know, I spent a long time at Google. It was where I met all of my co-founders, and we collaborated on a lot of different projects throughout the space. Um, you know, I was actually in a developer experience. Org for a lot of it building, developer facing tooling. Um, and that's really what got me into containers really early and built Google's container registry and that, that gave me sort of a front seat view, um, for all of the the container space. It's how I met my co-founder, Ville, who is one of the original folks on Kubernetes, and we collaborated in that capacity. It's how I met my co-founders, Dan and Kim, who were in the serverless org, trying to build, you know, things around containers. And so, um, over the years, we collaborated on all these different projects, um, you know, in the open source space, in the security space, a lot of them, you know, actually are sort of have interesting ties into the product that we are ultimately building today. You know, I joke, uh, that, you know, it was a master plan. It really wasn't. And we just kept seeing problems that, you know, we were we were experiencing building, you know, on the open source ecosystem. And we'd look around and nobody had solved them. So we'd try and put out something that solved one piece of it. You mentioned Distroless. That was one of them. Uh, Dan Kim seeing Sigstore, that was another one. And, um, really like bringing all these pieces together, you know, into a solution. You know, these are sort of parts of some of the ethos around security, you know, internal to Google building everything from source, things like that signing, um, you know, a lot of these sort of security principles Google had built up internally, sort of they sort of got a head start because they got, you know, so severely compromised years and years ago in the Aurora hack that, you know, and there's a really great videos talking about it. I think it's called the Hacking Google Series that they released a couple of years ago. But, you know, I think that that sort of security ethos and that sort of internalization of those things and then the exposure to how folks were using open source and containers in the outside world was sort of an interesting intersection that we all sort of got to experience, where we got to see those security controls within Google. But then we also got to see how folks were trying to, you build large scale distributed systems externally, and we could see what was sort of missing. And we kept trying to sort of fill pieces of the puzzle that we saw missing as we built things out. Um, but really, you know, at a certain point in time when there's so many pieces that you need to put together and hold the right way in order to build a solution. Um, you know, you really want someone to help you do that, right? Um, because you can, you know, put them together wrong or hold them, you know, wrong. And so, you know, I think that, um, you know, Sigstore is a great building block for helping secure supply chain, but it's just one of the building blocks, right? Um, you know, doing things around minimal images. That's another one. Building from source. That's another one, right? Um, and there's so much software that goes into modern stacks that, like, you aren't writing, you aren't even really building. Most of the time you're just pulling it off of. So a base image from Docker Hub packages off PyPI or npm. Right. Like the amount of software that we actually write to ship modern applications is this tiny like tip of the iceberg, um, compared to what we're actually running in our environments. But there aren't great ways of sort of safely consuming that sort of 90 plus percent of your stack. And, you know, a lot of the sources of those aren't necessarily following all of those, you know, hygienic practices. Right. Keeping everything fully patched and, you know, leveraging, um, you know, up to date dependencies that have all of the patches applied and things like that. So, um, really, uh, you know, I think, I think that's a big part of it. And Sigstore was really, I think probably the, the capstone on a lot of those collaborations that really prompted us to, to start a company around this. And, um, you know, I joke also like, you know, you're starting a company, right? Like, timing is like so important, right? And I tell people that, like, if we had tried to start a company doing exactly what we're doing a year earlier, we probably wouldn't have gotten funded. And when we started it, it was actually relatively easy to get funding. But so what changed? Right. Um, supply chain security went from being this almost 40 year old theoretical problem to a practical problem. Right. SolarWinds happened. Right. And it happened in such a big way that, like, it's a household name now, like, I had never heard of SolarWinds before this happened, right. And my you know, the non techies that, you know, I, I talked to and not just because I'm at Chainguard now but you know they've heard the term SolarWinds. But why. Right. It's because it was such a pervasive and high profile breach that was made possible through a supply chain attack. Right. It wasn't because they got breached. It was because they got breached and became the jumping off point to attack much bigger targets. Right? And so, um, so, yeah, I mean, that the timing aspect of it was was definitely really important. And, you know, it's almost like we've been preparing for it and building this sort of master plan for, you know, ten plus years beforehand and then brought all those pieces together when the time was right. But like, that's not the only way that timing is important. You mentioned mythos and really the, the other one that I think it sort of parallels, that is the rise in malware attacks, right? That both of those things, right on the vulnerability side and on the malware side, coming together, right, are creating great, well, great tailwinds for us from a business standpoint. But like, we've had sort of a five year head start on building out a solution that helps to solve those problems. And, uh, yeah. So timing is such a timing and luck is such an important aspect of starting a company. But, you know, I think we had studied this space and a lot of the pieces involved for many, many years before that.

3. Attackers at machine speed

Dev Tagare: It's incredible that a couple of just like just to summarize a little bit. Um, so it was a conviction in the fact that, like, everything needs to be secure by default. You're obviously operating in the developer experience ecosystem at Google scale, not just internally but externally, with all of these products sort of being open sourced incrementally. Uh, my, my startup exits were very closely related to, uh, Kubernetes going open source. I was working on a serverless solution. Kubernetes was an open source. We were swiftly shut down. So the timing piece resonates, although I was on the other side of timing in that case. Um, and the third thing probably was you wanted to connect the dots and go out of the door at the right, sort of like juncture, uh, to basically take Sigstore further. Now you're into the tailwinds of the mythos era, so to say. But Really? The agent took care of her. So, like, vulnerabilities and malware are coming together and they're I think of them as the jujitsu function where each is trying to, like, overpower the other. And somehow we're trying to, like, balance it. Um, so it's super surreal to see sort of that journey pan out, uh, in your career. And, you know, with regard.

Matt Moore: One of the visuals that comes to mind for me is like a canyon, right? And, you know, there's two walls to the canyon. You're trying to, like, fly through it, right? If you think about the world, like a couple of years ago, right before it really before we started the company. Right. Malware attacks weren't really that big of a thing. Right. And, you know, before things like mythos and some of the frontier models that exist today, it took a ton of skill and time for human to craft exploits for vulnerabilities. Right. So the window of time you had to to patch your vulnerability was like really long, right? And the risk of picking up the latest version was relatively low because malware wasn't really a thing, right? Malware attacks have gone from like, you know, almost very rare to, you know, quarterly, monthly, weekly, daily now multiple times a day. Right. And so this wall of the canyon sort of coming in this way, and then you've got the other wall collapsing too, where like it's now, you know, people are able to leverage agents to weaponize vulnerabilities at machine speed. And so you don't have this like mile wide canyon anymore, right? You're trying to navigate this canyon where like, it's almost damned if you do, damned if you don't. So you need a safe way of picking up those open source dependencies that helps you navigate that increasingly narrow canyon.

4. The open source trust model

Dev Tagare: I mean, like a good anecdote there that I see now in the open source is, folks, you know, like two years ago, three years ago, wanted n minus one versions of things because it was perceived to be stable. It was perceived to be like, you know, battle tested, if you will. And now folks are like, when is the next thing coming? Can I do like an alpha? Can I do it like a pre-alpha of this, which is, I think, a good directional indicator in terms of like what's going to come and how fast it's coming. Um, if you're releasing six weeks out, it's almost too late now. You got to be like three days out or five days out. Chainguard's position is that open source trust model is fundamentally broken. And it's it's like not just from like a funding or a staffing perspective, but like structurally broken across many tiers of the ecosystem. Uh, that's a pretty strong claim overall from like someone who spent over a decade building this infrastructure to, you know, fix a fix it. So when you landed on the framing of this problem, what really pushed you to go and tackle it yourself, as especially as the agent tech arc was panning out with frontier models and such.

Matt Moore: I think I think you said something important there, right? Which is the tiering, right? Like open source isn't just a single thing. And so I think how it's broken segments a little bit. But each of those segments, I think is broken in sort of different ways, but still sort of fundamental. Right. And so, um, you know, take the sort of legacy distributions model. I think they actually got a lot. Right. Right. So, um, the artifacts that they distribute are signed and those signatures are verified by package managers. The package manager database is effectively a bill of materials. Um, but the way legacy distributions are curated doesn't really scale or keep up with that accelerating pace of software development. You sometimes get that like older version thing. Sometimes it's way too old, right? You you don't get that ability to pick up the latest versions. Um, and I think, you know, one other aspect, you know, that's involved there, right? With respect to like, you know, the fact that these are signed and whatnot. You know, if you look at the barrier to becoming something like a Debian maintainer. It's remarkably high. Like. They meet up in person. They do like, you know, a key signing ceremonies and stuff like that. And ultimately it's because they gatekeeper access to the things that get signed by the distros release process. Right? Um, but I think some of the, the frictions around the barrier to getting something into a distribution and the pace at which they pick things up, there's also things around like supporting multiple versions and things like that. But those frictions, I think, are some of what gave birth to the language ecosystem package managers, which don't really have these. I think the first time I saw Maven, my first thought was, hey, that's like apt get. But for my Java dependencies, right? It's very similar, right? But it's different in some pretty fundamental ways. Right? It removes some of that. Like you don't get the latest version, you get whatever version you want, right. Um, but um, but I think that comes with the trade off

that anyone can publish anything. Right. And so you don't have that like carefully curated set of maintainers. A little bit after that, this you know, you also had things like Docker come along. Right. Docker hub. And it did the same thing for publishing entire containerized applications. Right. Um, but I think both of these had, um, you know, the same sort of failing. There's no way to verify the integrity of what you're consuming from these sources. Right. And even if you could write, like, imagine authoring policies for each of your npm dependencies, right? Because each of, you know, application written in node, you have like pulls in thousands of dependencies. Right. So if you need a policy basically helping you ensure the provenance of each of those things, and supposing hypothetically, all of those things were signed, the number of policies you'd be dealing with. You you just drown in. Right. And so I think what we've actually built out is, you know, in some ways effectively a hybrid of these models, right? We have the sort of cryptographic integrity and bill of materials facets that you get from a distribution. But we also have the virtually infinite breadth of the more free form package managers. Right. And we tend to cater towards the form factors that you would want to consume things in. Right. So if you're consuming system level packages you do that through the system package manager. If you want to consume language level libraries you consume things through those. But all of those artifacts have, you know, cryptographic signatures, building materials, provenance, and you're not verifying, you know, thousands of signatures. You basically are just verifying, hey, this was signed by the Chainguard factory. And, um, and that's pretty much it.

Nancy Wang: And so you move essentially then the route of trust, right, to Chainguard itself. And part of, you know, the secret sauce that I've always wondered about, Chainguard is there's so many different versions, right. Even if you take, for example, Python as a popular language, right? There's many different flavors. Forks, right. Versions. Um, how does Chainguard keep up? Right. Because I'm assuming not all enterprises are always operating off of the latest version.

5. Automation and vulnerability context

Matt Moore: Like I said, this is one of the struggles with, um, sort of the traditional distribution model. Python is actually a perfect example because most distributions have what they dub the system Python, because a lot of system utilities get written in Python. And, you know, there's, there's sort of, um, you there, it's often way behind. Right. So the list project I'll use as an example, we built around Debian packages, and this realist supports one version of Python. It doesn't support all the versions of Python. Why is that? It's because Debian has a system version of Python, and that's what we shipped. Um, at Chainguard, we basically track all of the upstream supported versions of a piece of software, and we build those and packages those as part of our distribution. So we have not only, you know, the, the, you know, older but still supported versions. But we also have the latest versions and uh, part of how we are able to stay on top of all of that is through extensive automation. Um, we've, you know, as a small, scrappy startup, you know, um, building out a Linux distribution with a much smaller engineering team than, you know, most traditional distributions. I think the only way we could really compete was through significant automation. And and so, you know, I think that now we talk a lot about, like I mentioned, attackers moving at machine speed. Right. It's no longer a human trying to weaponize vulnerabilities. It's, you know, um, machines doing it as models. Um, we, you know, through our automation, we've we've basically been making patches available to our customers at machine speed since before agents existed. And now that they do exist. We're trying to find ways that we can leverage them to take our automation to new levels, right. Can we make it more effective? Can we help leverage the agents in order to, um, you know, uh, take something that's 70% effective and make it 90 plus percent effective? Right. Because as we grow our inventory, right, that 30%, it becomes enormous. Right. And we need that to keep shrinking over time. Um, in order to, you know, remain efficient. Right. And so I think the answer to how we how we scale and how we do all this is significant investments in automation, what we call. You know, I mentioned earlier, signed by the Chainguard factory, we sort of jokingly call this our Chainguard factory. Um, it's a nice metaphor for this sort of assembly line of automation that we have, um, you know, taking things from source through those different artifact form factors and ultimately delivering those to customers.

Dev Tagare: So I had this, like, interesting, uh, paper I came across or like, it was really a stream of thoughts in one of my friend circles. It was essentially a concept where you, you know, you pull in a bunch of package management dependencies. You sort of do a builder pattern akin to the lines of like what Matt was suggesting with a, you know, Chainguard factory style. So you load up, let's say, ten different packages and different combinations, each of them individually certified. And now there's this emerging concept of like vulnerability chaining, which none of the scanners can actually trace. So there was like some cryptographic proof coming out that you could assemble each of these individual building blocks in the most secure way. And there's, you know, lowest common denominator, which is one, uh, package manager derived package or like one package individually. It is great shipshape, good to go, stamp, sign, etc., etc. but the combination is not. And then you go across n number of these combinations and you figure out what the vulnerability chain is going to be. So I'm just curious to get your thoughts on that. I don't think there's like an in-state solution yet to this

problem, but conceptually I could keep running this analysis and these, you know, building these scanners until something breaks eventually.

Matt Moore: Ultimately, I think context does matter. Um, for a lot of things, right. Some of it may be whether a vulnerability, maybe a critical vulnerability like log for J or whatnot. If you're not logging user source strings, you might not have been susceptible to log forge. But so many people were, uh, that, I mean, it.

Nancy Wang: Was, uh, hiking on a mountain.

Dev Tagare: There's a lot of trauma with that one. Yeah.

Matt Moore: Yeah. So, like, but, like, you know, so context matters. It's why I think a lot of the sort of, um, you know, graph based, attack based things like, you know, your Wiz, your orcas, your Upwind, etc. are super interesting. Especially like if you're dealing with, you know, an environment that's not running Chainguard where you've got, you know, a mountain of these vulnerabilities. Which ones actually matter, right? Which ones, um, you know, are exposing you where you have no mitigating controls in place? Um, and those are the those are the, you know, five alarm emergency things like, yes, you need to go and patch those things now. Um, but I think some of those other things, right, are things you should probably still patch. You may have a mitigation in place where, like, the tooling hasn't found a hole in it yet. It doesn't mean there isn't a hole. It just means you aren't aware of the hole in it. Right. And so you should still patch those things. But, you know, it may not be a get someone out of bed on a Sunday morning and maybe wait for the next patch. Um, you know, a release cycle and release. The qualified patches, sort of on a normal cadence. But there are things where put in context, right? Um, uh, they may become more vulnerable. And, you know, I think, you know, to your point, right, when you start to combine things as a contrived version of that would be like, yeah, okay, maybe the default configuration of something like nginx isn't vulnerable, but with certain configuration settings turned on, right? You combine those two things. You take something that isn't vulnerable out of the box, and now it is. Right. So, you know, configuration vulnerabilities are, you know, pervasive, right? Um, you know, so.

Dev Tagare: Yeah, it sort of elevates the problem to like a higher order abstraction where you're no longer dealing with like the fundamental building block or even a self-contained unit. Now you're like straddling the universe, which is super interesting to solve. And yeah, I definitely.

Nancy Wang: Say it really expands your surface area for attack. And so, yeah, speaking of attacks, right.

Dev Tagare: I'm in the right audience today between the two of you, I'm sure you're going to solve it.

Nancy Wang: Well. And you as well. Right from from the.

Dev Tagare: I'll try from them all the sides.

Nancy Wang: Yeah, exactly. Um, so, you know, we've obviously read on the news about, like, the xz breach the LiteLLM breach. Um, is there like, you know, based on, of course, what you know about sort of the attacker mentality. Like, how do they select, for example, what, you know, package to go and exploit. Is it because they get downloaded the most. Is it you know what what sort of signals do you typically track before you're like, hey, this is where I need to go. And and patch.

6. Supply-chain attacks and credentials

Matt Moore: I think volume is a big one that makes for nice targets. Um, you know, that's not to say that that's the only path, right? Like, so there's, you know, it's like fishing versus spearfishing, right? Like, if you're going after a particular target and you find out they're using one particular dependency, they may be the only person using that dependency. Okay. Go after that. But I think a lot of the ones that we're seeing show up in, you know, in the news, right, are these incredibly high profile things. And if you look at what a lot of folks are doing, you know, as part of this malware, it's stealing credentials. Why? Because credentials let you launch the next wave of these things, right? And so, you know.

Nancy Wang: And that is why you should never put credentials on disk.

Dev Tagare: Yeah. Oh, yeah. It was almost too easy. Almost. Thank you.

Nancy Wang: It's a layup.

Matt Moore: One of my hills to die on. Don't get me started. I yeah. Completely agree. Right. There's so many there's so many mitigations you can use to, like, not have this be a problem. But, you know, if you find a juicy target where you can, uh, like, run a crypto miner or something, maybe you get someone's AWS credentials. Um, you know, that's one way of potentially monetizing it, but you may find that you get access to, you know, um, some other target that may be

interesting. Right. And so by, by casting this wide net, by going after these high volume targets, one you obscure who might be an interesting target to follow up on and to. Right. Like it gives you I mean, it's it's like starting a company, right? You try lots of things. You throw lots of things at the wall. You try and figure out what sticks. And so by going after these high volume targets, you sort of figure out who you can go after next. And so it's a very appealing sort of smash and grab way of like launching a campaign that ultimately may lead somewhere. And the higher the volume, the more likely it is to sort of pay off. Um, but yeah, I mean, I we do see. AI. Related packages being targeted at a disproportionate rate. I think I think that just ties back to volume. Right. It's a lot of folks are worried about, like, AI specific security. And I think that in a lot of cases, um, you know, they sort of conflate these things where it's not actually anything about the AI that like, is making those the target. It's actually the fact that AI is so hot and the volume of those is so high. AI makes great bait for these kinds of attacks. Right. And um, and so, you know, I do think AI security is important, but I think that a lot of the attacks we see are baiting the hook with the incredibly popular packages like light.

Dev Tagare: Yeah, I completely agree. I think it's a I have a slightly contrarian, believe not fully contrarian to this, which is actually a bunch of these coding tools are making attacks incredibly easy to almost like Spearfish to mastermind, but also like cast a wider net and like keep casting, keep casting until you can, you know, go deep into one area with any number of customizations that you may see. So question for you, Matt, then is how much of an acceleration are you seeing in these attacks. Because I could get the the next AI coding tool and like keep doing this. Change the model, change the harness, be at it or run ten in parallel.

Matt Moore: Uh. You know, it's a productivity tool, right? It makes us more productive building software. Malware at the end of the day, is just software, right? So yeah, it's going to. It's going to accelerate it. But I think the other thing, right, with the acceleration in malware is it's sort of twofold, right? One, it's also relatively new. And as they're figuring out techniques that are effective, they're refining their techniques and doing things like, you know, I think one of the ones we saw over the last year or so was Shai Hulud, right? Shai Hulud was one of the first pieces of malware that self replicated. Right. So, uh, it it ran an install hook. It looked in your environment, you know, you didn't have to activate, you just had to install it. And then the install hook runs, it looks in the environment it's running and it's like, hey, you have an npm token, right? Uh, it looks at what packages you publish and it downloads them, injects the install hook and publishes a new version of them. Now everyone installing your, um, your packages is now susceptible at install time to that same vulnerability. And so that self propagation is a new technique that, um, uh, you know, makes these things, you know, orders of magnitude more effective, more worse. But these are the kinds of things where, um, you know, as far as that sort of attack path matures and folks get more sophisticated in it. Um, you know, they start to do new things, but also the agents just allow them to do that so much faster that it's, um, like I said, it's multiple times a day now, um, these things are landing.

7. Proactive defense versus reactive scanning

Nancy Wang: And so how do you catch them? Right. Because, you know, we talk about a lot of scanners, but most scanners really just scan for known CVEs or known malware. Right. I come from also the world of building, you know, data protection. Ransomware was also I mean, it's still actually even more a thing now with agents. Um, so how do you catch these, um, for example, zero days before even known that it's a CV, especially if it's burrowed in. It might be even dormant for a while before it starts releasing attacks.

Matt Moore: The time delay is a fun, fun technique. Um, um, and it's one where, like, a lot of folks are just like, oh, okay, I'll just not pick up a package for, you know, uh, a week or two days, like a cooldown period a week, two weeks, whatever. Wait for the outside world to figure out whether or not it has malware, and then I'll consume it. Right? Um, but to your point, right. Like scanner, scanner is fundamentally, uh, I view as a sort of reactive technique, um, because you need to know, like a malware signature or you need to know that a piece of software has a particular vulnerability. Right? I look at the way we do it as, um, sort of proactive. Right? We are. We are doing things we're sort of operationalizing best practices at scale in a way that eliminates entire classes of vulnerability. Right. So by so one for instance, is we build everything that we distribute from source. And that turns out to be extraordinarily effective and mitigating malware attacks, because most of these malware attacks aren't actually going after the upstream source repo, they're going after PyPI or npm. And like getting a developer's credential and just publishing a new version. That new version, you know, may claim to be, you know, based on certain source code, but, you know, it injects a backdoor or something like that because binaries, you know, they're opaque, they're hard to inspect. Right? And a lot of people just trust that, you know, that correlation, you know, is is strong. We aren't relying on those artifacts that contain the malware that those those attacks are something like 99, 98% of malware attacks that happen are going after those intermediate things that folks just implicitly trust to be the source. Right. So by by building things from source. We sort of proactively eliminate huge classes of vulnerabilities. And so that's one thing we do. But to tie it back to we mentioned late Elm a second ago, right. Think about late Elm. Right.

Late Elm was compromised 100 million installs or something like that. Close to 100 million is mind boggling. You know, volume to your point about who gets started at.

Nancy Wang: Definitely quite a few startups we know.

Dev Tagare: I know, yeah, it was rough. It was rough for a few weeks.

Matt Moore: And not startups, but yeah.

Dev Tagare: Not startups.

Matt Moore: So late Elm was discovered because it was using more memory than a developer, you know, was expecting it to and dug into it and discovered the malware. It's actually remarkably similar to how XSS utils was uncovered a couple of years ago. Right. There were, you know, some some performance issues. And, you know, G10 was unmasked as someone who was actually trying to launch Um, a very sophisticated attack. We as an industry got lucky that it got caught so quickly. Right? Um, and because it got so quick, caught so quickly, the scanners could update their signatures, they could catch it, they could find all the places it was running. But the reason our customers weren't impacted wasn't because we were, like, better at scanning. It was because the attack fundamentally, um, you know, published a version of npm that we wouldn't build. And so, um, because we build everything from source and the source containing the malware wasn't, you know, actually published, actually available. We never packaged it. So our proactive approach basically avoided this entirely. And suppose we hadn't gotten lucky. We we as an industry hadn't gotten lucky and it had gone, you know, weeks, months before it got discovered, our customers still wouldn't have been susceptible. Um, but everyone installing even running malware scanners potentially would be susceptible. And so I think that's, you know, I did this Secure by Design initiative where they talked about eliminating classes of vulnerabilities, things like using memory safe languages, things like using, you know, ORMs to avoid SQL injection. Right. I view building things from source and consuming from a trusted source. That's maybe doing that for you. Like us, as eliminating entire classes of vulnerability, like trusting those intermediaries, which, you know, as I mentioned earlier, that fundamentally broken, you know, bold statement. Um, you know, I think that it's that implicit trust in these intermediate artifacts, which, like I say, like I said, sort of it's such an overwhelming number of malware attacks that tie back to that, that it is fundamentally broken. And even if folks were signing each of those things, you would be drowning in policies to verify. And what that means, as I think we all know, is people won't do it if they need to author thousands of policies or they'll be really weak policies that may let things through.

8. Agents and the software supply chain

Dev Tagare: So somehow one of the problems that I've been thinking through is like the act one was sort of, you know, move away from these YOLO binaries and package managers and such that you download off the internet, or let's say, one of the curated hubs on the internet to something like a curated package manager, a gene card of the world. Act three, in my mind, is coming, coming to to the fore now, which is going to be agents generating code against a harness. The harness is effectively moving and shifting every few weeks. And and you're running in some base environments, let's call it like certified, you know, binaries. Everything comes stamped by the factory, so on and so forth. But the new code that's generated is effectively, you know, just directly executable code. And nothing is being built in like an image format in the canonical sense by it. And that new code is also mutating pretty aggressively. Pretty fast, as you know. Um, as the world moves to the world of, like, prolific AI, as I like to describe it. So in that world, how do you see your company's role in your role evolve, and how are you thinking about solutions in that space?

Matt Moore: There's a couple of different ways that I think about it, right. So like agents building software. Right. Like there's sort of two paths, right. They can reach for preexisting packages and and leverage them, in which case it's just like the sort of, um, model we have today. It's just instead of humans installing packages, it's the agents installing them. And they need a safe way of consuming those things. I think the other actually worries me a lot more, not for Chainguard, but for, um, because it sort of breaks the security model. Right? So let's say hypothetically, instead of reaching for a particular library, um, the agent which has this encyclopedic knowledge of software, um, recreates in your tree. Um, it's effectively the same functionally, but, I mean, you have a lot more code to maintain and hopefully review, but, um, but you know, what do you lose with that? Right? Um, well, you lose the fact that if it was trained on a version of that software that had certain vulnerabilities, scanners aren't going to necessarily match those things together. Right? Um, you know, they won't they won't know you're susceptible to CVE 2026, blah, blah, blah, blah, blah. Right. And so I think that. It. Behooves us to actually continue to tie into ecosystem dependencies, um, so that we can have that awareness. Um, you know, even with things like SaaS scanners and, you know, the agents themselves getting better and better and better, um, you know, it's

you, you run the risk of like, well, what are you missing by effectively recreating that software? And, you know, you're, you're signing yourself up for, um, you know, the upkeep of that software.

Nancy Wang: Thinking about the maintenance.

Matt Moore: Yeah. Exactly. Right. Like who? Who? You know, like I said, the 10% or the 90% of that sort of iceberg, right? Like, do you want your do you want to pay for your agents to basically be maintaining ten times as much of the code, you know, day to day and, you know, cleaning up tech debt and and fixing bugs? Um, you know, like, if you can sort of focus on that, that top layer and have the agents help you, you know, stay on top of those latest dependencies. And really, it's, you know, picking up like breaking API changes. You know, this version has another parameter. Okay. Can the agent help you fix that up? You know, just think about it like dependable to renovate plus plus. Right. How can it. You can really augment that. I think that's a much safer world to live in, even though, like, you are potentially consuming, um, dependencies from, um, you know, someone that's not you. But I think that's sort of a feature, not a bug. You just want to make sure that you can trust the the stuff that you're getting, I think, which is where a company like Chainguard, I think fits into it.

Nancy Wang: Yeah. Actually, let's talk about some of the maybe integrations that you've built right, with Cursor and Cursor. Maybe that's also where you're headed because um, and maybe tell us like what let's say your plugin does when an agent's trying to, let's say, scaffold a new service and is releasing reaching for like a base image.

9. Secure images and agent environments

Matt Moore: You have, uh. A few different MSPs that sort of make our inventory, you know, discoverable and available. But like, I think one of the main things I touched on this a little bit earlier, we try and expose artifacts in a way that's sort of, um, compatible with how you consume them. Right? So if if you are using Maven for development, you want to be able to pull Java dependencies in a way that, like Maven works. And, and folks do this today, right? They don't necessarily pull directly from Maven Central. They may shut off, you know, access to arbitrary dependency sources. They may still be pulling from them, but they may be pulling through something like, you know, a corporate managed artifact or instance or Cloud Smith instance or some other intermediate repo where they can shut off the outside world, but then proxy that through an artifact manager. Right. So there's already great facilities for having a tool like Maven not consume things from Maven Central, but consuming it from your artifact or your cloud mimic. Um, the same for, you know, Python dependencies, the same for, you know, go and the go mod proxy, etc., etc.. So every ecosystem has a way of sort of, um, not pulling from the, the open internet, but pulling through some sort of artifact manager. And so what we try and do is we try and make the artifacts that your users are consuming, um, sort of have a compatible source. So, folks who are consuming our Java artifacts can basically, you know, instead of pulling from Maven Central, they pull from libraries. Dev dev is our registry domain. And and they can pull Java artifacts from their Python artifacts from their node artifacts from there. So libraries are largely drop in. Um, it's a little bit different when you start to talk about images because, you know, depending on the image you're building on, there may be subtle differences. So, um, you know, one example is like, forget about Chainguard for a second. Say you're switching between, say, a Debian image or an alpine image. What's, what's one of the things everyone does in a Dockerfile? They run the package manager to install some more packages. Well, it turns out, you know, Debian based images use a different package manager from alpine based images, use a different package manager from RHEL-based images. So I think some of what you see when you're like, uh, switching to a Chainguard based image is, you know, a lot of a lot of the sort of delta in terms of migration is really just those same sorts of changes, right? So we we use the APK package manager, um, distributions like Debian and Ubuntu use the Deb and APT, get package manager distributions like Raul Souza, Amazon Linux, etc. use, you know, RPM and the yum ecosystem. So um, and then alpine which we, we use the same package manager as alpine. We, we use the APK based one that um, alpine also uses. So it's sort of segments by like the package manager ecosystem. But like beyond that, you know, oftentimes Java is Java. Now it's much more nuanced.

Nancy Wang: So what happens when like an agent reaches though for, let's say, an image that you all don't have? Um, I mean, you all have a lot. I think it's something like almost 3000. Right. Hardened container images, millions of libraries. So probably a lot of ground. But what happens is that edge.

Matt Moore: Case one and when humans don't have it, there's a way now through our console to go and request a new image. I don't think we've exposed a way for agents to be like, yes, my user needs this image yet, but that's a good feature. I'll talk to our products team about it. Um, but I think one of the things, you know, agents are problem solvers, right? And so I think one of the things that we see is, um, they they may not there may not be like a ready made image that does what, um, you know, the agent wants, um, maybe it's maybe it's something like they want Python, but they also

want these, like, additional packages, right? They could start from Python and install those additional packages, but they can also potentially start from our sort of, uh, Wolfi base or Chainguard base image, depending on whether you're using free or paid image and then install additional packages from that. And we have a large library of packages, some of which we've just been asked by customers to build that package so that they can install it. And so we may not have images that cover all of the packages that we have. But that doesn't mean we don't have it. So they may be able to construct their own image from it, either as a combination of an image we have and certain packages or just those packages. And then in the event that we don't have a package or an image, there's a path to file a request with us. Um, right now, humans only. But, um, but maybe in the future we'll provide like an MSC endpoint or something where the agent, maybe on behalf of their user, can request, um, additional images or packages.

Dev Tagare: Yeah. The way I'm thinking about it is if you don't have the package one, your options are don't allow it to, you know, call Matt up, uh, effectively. And they give you a feature where you can, you know, if you have the source, bring the source, we'll build it for you. We'll certified for you if you don't have the source. You just want to arbitrarily bring in something. Give it to us. We'll run the end. While not really scanners we have at our disposal. Build a effectively equivalent of a sandbox for you with those packages and dependencies installed already, and run your code in it. And the extreme end of it is like, okay, you don't even want my sandbox. This is my assessment. You know, feel free.

Nancy Wang: To.

Dev Tagare: Take the risk. Yeah, you take the risk. So I see like those three as potential paths maybe.

Matt Moore: Yeah. And there's a lot of reasons we might not have a package. Right. So what I mean, one is it might not be open source. Right. Like I think the funniest one is if folks have asked us, oh, can you build like windows images. It's like, yeah, as soon as Microsoft open source windows we'll start talking about it.

Dev Tagare: Yeah.

Nancy Wang: Yeah, yeah. I mean I think you live near.

Dev Tagare: You and or get a liability insurance quickly.

Matt Moore: Yeah. So we do we do have something that we call our commercial builds program where we are starting to partner with vendors, where maybe, maybe it's the source is available, but it's not licensed in a way that we can build and distribute it. And there are some where it's actually closed source and we partner with them, they give us access and we build and distribute it to mutual customers. Um, but um, but yeah, I mean, we also build stuff that folks want to sort of filter out from their users because, you know, they may be using a license that allows us to build it, but they do not want in their environment. Right. So network copyleft is a common one. Um, you know, things like agPL, there's things like us LCL. And so, you know, that's we try and give folks the option of having a clean source for all of the software that we can possibly build, um, you know, including now, not just open source, but in some cases, um, commercial builds and, uh, not technically a OSI approved license, but source available things. Um, but, you know, that doesn't necessarily mean that enterprises want to make all that whole thing. Especially now that we have ways of entitling customers to our entire catalog that they want there. They don't want their developers, you know, accidentally putting them in a difficult position with respect to license compliance.

10. Agentic code review and CI

Dev Tagare: Recently you published an article that was like, you know, Chainguard uses a fleet of agents to do effectively what I would describe as SDLC and enforce standards across your code base. You generated a bunch of like peers and closed and auditors, like 75% of it. Um, you know, without any human intervention. I've been trying something similar with ADK, which is our, like, open source agent development kit. You have a bunch of issues on it. You take effectively the equivalent of Matt's fixer. Go, you know, generate code, commit TLS, and off you go with tests and merges. Right. So for an engineering leader, um, you know, that number is first staggeringly impressive 75% growth rate without human intervention is massive productivity gain. But it's also kind of alarming on the flip side, right. Because now you have no human intervention over, let's call it 75% of your bugs or vulnerabilities or, you know, potentially a feature creeps, so on and so forth. So how do you build the confidence level in your, you know, agents or in your SDLC agents to really let them rip without any human intervention?

Matt Moore: So I'll, I'll start with, uh. Sort of one, one piece of it. Right. So I think the, the first piece is we've invested a ton of effort to try and make it so that if CI is green, it's a change is safe to merge, right? Um, you know, this includes idea things like agent reviews on, on a PR, um, whether it's authored by a human or an agent. Right. Uh, you know, the goal is

green means merge. Red means, you know, don't merge this. There's something wrong with it. Right? Um, and, um, forget about agents for a second and think about the productivity boost for a team. If, as an author of code, the Machines give me all of the feedback that I'm going to get as part of that, like pre submit process in a matter of minutes. Um, one of my favorite horror stories about this was, uh, a Google. They had this readability process, and at the tail end you would you would add this group who were like specialists in go to review your code. I, I'm sure you're familiar with this.

Dev Tagare: I am I want to say so many things about it, but I'm like, uh, you know, I, I want to give the editors less things to do.

Matt Moore: The last few reviews I had leading up to getting go readability, I remember it kept picking the worst possible reviewer time zone for me. And so it was this, like basically 24 hour cycle time, right? And so imagine if you can get all of that out of the way in seconds to minutes. Right. And like that is a huge productivity boost. Right. And then as a reviewer, the automated checks, you know, already cover all the pertinent feedback that you might give. Right. And so you're sort of, you know, spot checking things at the last minute. Right. Like that's a huge productivity boost, both for me as a reviewer as well as for people authoring code. Um, I think that when you start to talk about an agentic review, there's sort of an interesting mindset shift that sort of needs to happen. And, you know, we're getting, you know, bigger and bigger changes, but also more and more code reviews. Right. And so I think the mindset shift is whenever you're leaving a code review comment on a piece of code, you sort of have to ask yourself, how can I make the automation catch this sort of thing everywhere? Right. And you start shifting from sort of reviewing things by hand to authoring policies that can be enforced universally, not just on the reviews you happen to do, but like every review that goes through the system now gets that consistent level of enforcement, right. So so sib ingredient is is one thing right? But I also like to think about it a lot like defense and depth right. You want to have a lot of different layers of of protection with redundancies, especially when you start to talk about things like auto merge. And so when you start to talk about automatically merging things, right, the complexity of the changes matters, right? Smaller, more focus changes are better. And I think that does sort of inflate the number, right. Um, a lot of the changes that we produce with our system are intentionally very focused so that they can be, you know, I've had this mantra for a long time, right? Like, if you're sending out a code change for review that has multiple things going on, split apart the contentious parts from the non contentious parts, and you're going to make your life better. The non contentious parts will move while you go back and forth on those contentious boards. Right. So by. By segmenting these things. Right. And you have lots, you know, a high volume of really non contentious, very focused changes. Um, you know, low complexity, small impact. It enables you to, um, you know, get those things through more, more quickly. So, um.

Nancy Wang: I think you just gave our listeners a rule book for, uh, jacking up your, uh, PR account.

Dev Tagare: I know, yeah. I mean, one of the things I've been like, anecdotally, looking at Matt and Nancy is, um, with effectively loops becoming a thing. Now in coding, everyone's pushing like 1000 line sales and PR and I'm like, Jesus Christ, you're mixing too many things together. And the response is, you know, I'll add a flag to protect it. I'm like, now there is a flag proliferation. Like, what is happening here? What happened to single responsibility? Like, don't mix ten things together.

Matt Moore: And agents are shockingly good at taking a big change and breaking it down, right? Like so. Break it. Break it down into a progression of three, five, ten commits. Right. Give me a branch for each of them. I'll push each one. Iterate. Land that rebate and it rebase the stack to. I mean, I think one of my one of my favorite things to never have to do again is deal with a merge conflict on go mod files. Going through and removing all this stupid little, you know, less than greater than equal signs. I can't even remember the last time I.

Dev Tagare: Had lost memory of that, actually.

Nancy Wang: And you blocked it out of your memory.

Dev Tagare: I had to be honest. I'd forgotten about it. Like. And I write go, like, every day.

Matt Moore: Yeah. So, like making that go away, like, huge. Huge. Right. And. And so how have the agents break down the change, you know. And I, you know, I wish I could have it like, even potentially push and merge things, but I actually have it set up so that my to, to push to get and to fetch from private repos. My my SSH key is my hardware token, so it can't push as me, but which I actually really like from a security perspective. But, uh, it makes it, you know, a little tedious. So now I have to actually do the pushes myself and whatnot, which is, I think, a small price to pay for that balance. But but yeah, have the agents break down your changes. Um, and, you know, when you're having agents write code, have them segment the changes into things that are focused non contentious. Um, I think some of the other aspects we look at are

things like how sensitive is the code it's touching. Is it production configs. Is it dealing with auth. Is it things like that. Um uh you can also look at things where like the, the changes touching both the code and tests. Right. The tests may be protecting against some sort of regression. And you know I agents love to be like, oh, that that test doesn't matter. Skip the test or delete the test. It's like, no, like don't don't delete the test. Um, and so, you know, those things I think are a signal of, you know, one potential risk. And, you know, we do all kinds of, you know, an agentic code reviews, you know, the tool, the tools, like Claude. And if you put it in ultra code mode or whatever it start to do, like adversarial reviews and stuff like that. And so I, you know, I actually find that running a genetic reviews, especially of complex changes where partway through your eyes glaze over, you tune out a little bit, are significantly better at sort of rocking the full, change some of the context behind the change and giving you like, meaningful feedback. Um, especially where it touches on, you know, standards like OAuth, right? If you're dealing with an OAuth flow. Right. Um, you know, it has this great knowledge of, you know, standards like OAuth and OIDC and, you know, and on and on and on. And so, um, you know, things that, you know, we were talking about, like reducing the skill barrier for exploiting things. I mean, I think the same applies to some of those other things. Right. Can you get really actionable feedback out of something that just has been trained on some of those open standards? So it's a lot of things, but yeah, that's a sampling of some of them.

11. Identity, credentials, and least privilege

Nancy Wang: I feel like with you mentioning SSH keys and then following with OAuth and OIDC, I feel like you're just teeing me up today. Um, so, you know, now, of course, I've been waiting to ask you about this, so this doesn't feel like a product placement for 1Password. But I have to say, you know, tell me, tell us more about your mantra of, you know, the hill that you want to die on is no long-lived credentials. And look, you've built your own STS service for GitHub Actions, right? So, so tell us like, um, I guess how come you're not at 1Password?

Dev Tagare: Um, I.

Matt Moore: Love 1Password. I like not not to not to shill your product. But I love 1Password and I think that, you know, I not. Not that it's in the sort of short-lived credentials realm, but like, I think that it really sort of grinds my gears how often. Like we were talking about credential leaks being the source of a lot of these breaches. Um, you know, I sort of look at things along two lanes, right? There's human identities and there's machine identities. Right. And I think along both of these lines, um, avoiding these kinds of things is a solved problem, right? I think for human identities, um, put all of your, you know, long-lived credentials in something like 1Password, leverage MFA and use so to sign into things. So you're minimizing the number of sort of providers that can deal with it. Um, you know, I love the fact with, you know, a tool like 1Password where like, it's not going to show me the magic little, like, fill in my password thing if I'm not on the right site. Right. And so for human identity, like MFA and password manager. Like, um, I basically can't live without them. Um, on the workload identity side, like Federation is a thing. It is available almost everywhere. And, you know, we in some of the few places where it's not available, like GitHub, like we built our own and we run it as a public good so that literally anyone can use our instance. Um, and if you don't trust us, it's also open source and you can host your own. And basically the way this works is it runs as a GitHub app, the app key. And I could rant about how you get that app key from GitHub, but the key gets loaded into the KMS system and basically, you know, never seen again. Right. And so, uh, and it has all kinds of like alerts around it so that if you look sideways at, um, uh, the KMS system, it will actually page someone at Chainguard. Um, but but yeah, like, I think those two things. Well, it's not a panacea, right. But those two things are so effective at mitigating a lot of the attacks that launch these kinds of things that, um, I it I don't know, to me, it feels borderline negligent that so many long lived credential leaks are leading to the kinds of issues that we have. And I even said, um, I think almost a year ago, I think I posted that one of the biggest misses of the SLSA framework was, given the abundance of long-lived credentials being used to compromise the supply chain was not mandating the use of short-lived credentials. Um, as part of, you know, the build standard, um, because they were like, ah, that's an orthogonal thing. I was like, ah, it's not really right. Like if you look at how all of these things are being compromised, it's almost always like GitHub leaks or npm token leaks or whatever, right? Like and you, you can federate to do something. It's not perfect, but it is so much better than, you know, giving an attacker weeks to months to whatever to to potentially figure out how to use that for maximum, you know, effect.

Nancy Wang: Well, given your passion for 1Password, this is actually where I'd love to get your thoughts. So the reason I actually, um, you know, led me to join 1Password is I, for the first time, discovered about two years ago that we actually have a full, uh, developer suite offering. So we, we store SSH keys, API keys, uh, paths, if you must use them. Right. And recently we actually extended our coverage now to just in time privileged access. So things around dynamic access. Right. Um, you know, rotations giving you access only for the tasks that you're doing with acquisition of Pono. So now we have an Israeli dev center, um, that we're excited about. Uh, so thoughts and we'd love to get your feedback.

Matt Moore: So it's sort of complimentary things, right? So like you're talking about like privileged access management where you sort of pseudo and elevate credentials and things like that. Right. So the lifespan of credentials is sort of complimentary but sort of orthogonal to like least privilege and what you actually have in terms of privileges. And I think as it pertains to sort of a genetic security. Right. Like humans accumulate a lot of credentials because we do a lot of things, and agents are starting to do a lot of things. Right. So we start to give them a lot of credentials. Right. And so it's we're in this sort of weird spot where it's a workload, but we're starting to give it a lot of credentials. And so I'm not sure we've adequately solved this yet for humans or agents just in general. But I think there's a number of tools, and I think you've touched on some of them. A super heavy one is like identity segmentation. I actually have two accounts at Chainguard. One is super admin and I basically never use it has a separate security key in all kinds of other stuff. Um, and then there's what I use day to day. Right. Like, I don't want to be running, you know, agents or things like that. You know, as my super admin identity, it runs as sort of a normal identity that can't, you know, do crazy things, right? Um, but even when you're using the same identity, you know, there's, there's patterns like OAuth where I can have some baseline set of credentials and I can sort of subset that to delegate some limited access. And then there's the sort of complement of that which you just mentioned, which is sort of the pseudo pattern of my baseline is actually more secure and, you know, smaller, but I can sort of go through a flow to elevate my privilege as sort of pseudo, uh, to do some sort of action. And so I really like that pattern. I think that, you know, on the cloud provider side, it took a while for some of the folks like GCP to really, uh, get some of that stuff in place. And I was a little disappointed with how broad my, you know, super admin credentials had to be in sort of a steady state. And like you talk about folks being on call and things like this where you want them to be able to take action and emergency, but, you know, you don't necessarily want their agent to like RMF, a database or something like that. So, um, so yeah, I mean, that idea of sort of elevating permissions just in time is definitely, I think puts things in a better place. There's one there's one other thing that we sort of have played with that I touched on a little bit with Octo SES example, but like we so think about least privilege and it's very identity centric. I have an identity. I give it the minimum permissions needed in order to accomplish it starts right. It's pretty straightforward. But, um, identity models have a lot of like inheritance and other things, especially when you're talking about humans. And so, um, we we put out a blog post a year or two ago with something that we call audited least privilege, which sort of flips that on its head. Right? So there's the identity centric view, but you can take a resource centric view. Um, and dev. Since you're at Google, I'm going to I'm going to complain a little bit about this, but.

Dev Tagare: Go for it. Go for it. Yeah.

Matt Moore: Basically what the way it works is you can take a resource and you can say, this is the normal access pattern and one Terraform resource. You can have, uh, a log based, uh, alert policy that goes over the audit logs for that resource. And anytime there's an anomalous access, it can alert. Right. And so this is how we protect the keys and things like octo ses. We started to layer this in everywhere, like Terraform modules that like are part of building more complex systems. We started to layer these things in everywhere. And we actually hit a system limit in GCP where we could not create more of these things and we could not get it raised. Um, and so we unfortunately had to start like pulling them out of a bunch of places that had a really high, um, you know, degree. And we were probably like, you know, debatably abusing this, but it's such a cool pattern to sort of flipping flip things on its head and say, look, this, this database is an implementation detail of like our data store. Right. It should be accessed in these ways by these identities. And if we see anything out of the ordinary according to the audit logs, um, send an alert and alerts. I mean, alerts like escalations aren't necessarily a bad thing. Sometimes it's the process just flagging something. If the if the person who's on call needs to break glass and do something, they do that sort of pseudo elevation. Um, great. That's that's audited. And, you know, they get the access they need to do what they need to do. And then if the actions they take send alerts, um, you know, you don't necessarily want to distract your on call, but it raises awareness that certain actions are being taken that are out of the norm. And you, you are at least aware of it. Um, my funniest anecdote with, uh, this stuff is, um, we. We did some POCs with some of those graph based scanners. I won't name names, but, um, some of when we were doing our evaluations with them, I don't know that they found any, any sort of severe things in our environment, but we caught every single one of them, like scanning, scanning our environment with our log based learning things. Because as soon as they were like, you know, like I said, if you look sideways at the desk, we we alert. And so just like listing the key versions, um, will actually alert us. And so, um, there were there were a bunch of operations, some of the, some of the, um, these tools will, like, create a backup of your database so that they can scan the backup instead of scanning your live database.

Nancy Wang: Side scanning volumes.

Matt Moore: As soon as you do that, you triggered one of our alerts. So, um, there are a variety of ways we sort of caught them with, uh, with this sort of notion of audited least privilege. Um, and, you know, I think these things aren't mutually exclusive, right? It's defense in depth. Right? You can layer in all these different ways of doing things, leveraging

hardware security keys, elevating when you need a certain level of access, and then auditing, continuously auditing, um, you know, how things are being accessed to make sure it sort of fits, fits that norm.

12. The future of machine identity

Nancy Wang: Yeah. Now I'm actually like, thinking about, I mean, probably taking a whole nother episode of things that we could probably do together because, yeah, we're we're making rapid strides on sort of connecting all of the, the human, the machine workload and agent identities because, you know, to your point, right. Agents sometimes do a lot of things that humans can do, especially now with computer use. Right. Taking over your browser. Let the point and click. I mean, that doesn't have to be just CLI anymore. I mean, this morning we just announced, uh, integration with uh, Claude Cowork. Right. So Claude Cowork can now do things on your behalf using credentials in 1Password without actually ever seeing the credential. Right. So that's more of a consumer use case, but certainly on the developer side, there's a whole like sort of git just in time, making sure that credential gets rotated or just, you know, gets cleaned up after the use. So I feel like there's a lot of sort of secure by default defense and depth, plus also the identity security that we can layer on top of each other.

Matt Moore: Yeah for sure.

13. Rapid fire: responding to a compromised package

Nancy Wang: Cool. So let's do our favorite portion of the episode, which is rapid fire. Um, so let's start with a package in the top ten or top 100 on PyPI gets a compromise tomorrow. What is the team running cursor agents need to do in the first hour?

Matt Moore: Uh, if they're using Chainguard libraries, nothing.

Nancy Wang: Great answer. Uh, one thing platform teams are doing right.

Dev Tagare: Now, and all they should.

Nancy Wang: Exactly. And they not they should. And Matt will give you a special signup link. Um, one thing that platform teams are doing right now that gives you confidence.

Matt Moore: I think waking up to the real dangers of complacency around CVE volume and [audio verification needed: phrase rendered as “the rapidly growing threat landscape”]. I was talking about CVEs and malware and the need for supply chain security in general.

14. What comes next for Sigstore and Chainguard

Nancy Wang: Yeah, awesome. And Sigstore is five years old. So where does the ecosystem need to go in the next five?

Matt Moore: I think normalizing signing artifacts and verifying their signatures and, you know, signing is one part of it, like they're signing it. And then there's a testing, which is like signing a claim about it. I think we need more open standards around the types of claims we make, like a SLSA attestation or an SBOM or things like that. Right. What are the types of claims you want to say about it? So expanding that ecosystem to cover more facets of things.

Dev Tagare: Where do you envision Chainguard to be like, let's say three years out and that that [audio verification needed: phrase rendered as “that gap closing”] is do you think you're going to widen that again, figuratively speaking?

Matt Moore: I mean, I think that hopefully for our customers like we we will navigate that canyon for them and they just won't have to worry about it. Like they'll happily ride on top of us. You know, above that canyon. Well, we we navigate the narrow, narrow, uh, canyon walls. Um, but three years is a really long time. And I think, you know, ultimately, we want to make we want to make navigating that canyon, which is getting more and more exciting as the walls are closing in. We want to make that boring, right, for our customers and may be exciting for us, but we don't want it to be exciting for our customers. We want it to be boring for our customers, because if it's boring, then they can focus on, you know, the differentiated business value they're building on top of open source. That top 10%.

Dev Tagare: Makes sense.

Nancy Wang: As we say that at AWS you handle the undifferentiated heavy lifting. So yes.

Matt Moore: Yeah. And I love the example of like, you know, okay, 25 years ago, what did it take to build a service? Well, you start by racking servers, right? Things like AWS made that a thing of the past. But the layers of abstraction just keep

going up and up and up and open source is helped with that within the application layer. But yeah, I mean, if you think about like everything, the whole of what it took to build a service 25 years ago during the.com, uh, boom, like it was, it involved racking servers. And how many people think about that anymore? More than more than probably should. But yeah.

Nancy Wang: So Matt, that was an incredible hour of just, you know, rapid fire reminiscing down memory lane. Um, but look, I wanted to also give you an opportunity to demonstrate to our audience here, you know, what Chain guard looks like in practice.

15. Chainguard in practice

Matt Moore: So, as you said, we've built almost 3000 different container images. These are backed by many different pieces of software that we have as system packages. And then we have even more libraries available to look at our containers, which is where we started. You can go to Chainguard. And so say I want to, you know, look at something that we have. So pick node super popular so you can see the different tags we have available node. We have a number of images that are, you know sort of free to download the latest version. Um, and but we have a lot of like I said, we build all of the upstream supported versions that we have. 26, 25, 24, etc.. Um, so one of the cool things that we added a while ago was the ability to sort of compare these with upstream images. So if I go over to the comparison tab, you can see this sort of contrast. So this is just node latest. Um the official node image uh on the right hand side. And we scan, um, you know, a variety of upstream images once a day. And then each, each bar in this chart is sort of a daily snapshot. So you can see the Chainguard one right now has two medium CVEs. Um, but the, you know, the the node latest image has 90 critical CVEs. It has a total of 822 CVEs. So a huge number of CVEs, if you just go with node latest and like if you look at the relative size of the images, um, our images is dramatically smaller. Um, so we do scan a bunch of them. So if I flip over to the node slim, it's a different picture. It's not quite as dramatic. Uh, many fewer CVEs, but still smaller. And, um, you know, there's still, you know, most of those. Well, I think it was 80 or so. There's still a lot of those critical CDs that are, um, uh, in that, you know, node, slim image. Um, so with respect to these medium ones, right. Like, I think one of the things I joke about is, you know, dealing with CVEs is a lot like alert fatigue, right? The more you have to deal with, the more you become desensitized and the the more you stop looking at it. Right? And so when you have, you know, 800 or how many CVEs that was, that was in one image, right then that bass will show up in your scans of every image you build on top of that. Right. Um, so, um, you know, when you have that high a volume of CVEs, where do you start? Right. Um, and so you have a lot of people look at it and they're like, well, I, you know, I'm, I guess I'm screwed. Um, but when you have two, like, when I looked at this, I was like, why are there two? And I actually dug into this and, uh, it's a medium severity CVE. It is patched upstream, but there hasn't been a release yet. And, you know, our engineers have looked into it and even looked at cherry picking. Um, but it doesn't cherry pick cleanly. And so we didn't. And we also saw that upstream was, you know, there was a release imminent. So, um, so basically this should go back down to zero very, very soon. Um, but, um, you know, the dramatic difference between, you know, even a hundred CDs in this with, you know, there's more critical CVEs in the slim version of the node image than, you know, in the latest version of our images. And so, you know, that's that's some of the contrast there. I think one other thing that this page tries to visualize is sort of the trend over time to one of the things that we found with sort of minimizing images is, um, the less things you put into an image, the slower it'll accumulate CVEs, because there's less things catching a CV that may show up in your scans. Right. So we plotted this sort of over time and this sort of accumulation of CVEs over time. And so you can see the total CVEs and then you know net new CVEs identified. Um, and so there's another cool view that was added, um, uh, relatively recently where you can actually, um, go to our directory. It was at the bottom of the page. You can actually enter a whole bunch of images and it'll show you sort of an aggregated view. So if we just go to this default, you can um, look at the here you go CV comparison. And so across this set of, you know, go node, Python, ruby, rust. Um, there's a handful of CVEs, you know, like that one we saw or the two we saw in node. Um, but there are across these five images, over 3000 CVEs, 400 critical CVEs. And so, um, you know, it's it's a pretty stark contrast. And we spent the first several years of, uh, you know, the company selling this product, convincing people that what we were doing wasn't, like either somehow lying or hiding stuff from the scanners. People didn't believe that you could go from this world of having, you know, thousands of CVEs to, you know, damn close to zero, if not zero. Right. And, um, and but, you know, I think the reality is we actually go out of our way to have this show as much stuff as possible. You know, there's there's this really fun, uh, test that, ah, my co-founder and CEO, Dan, uh, calls the WordPress test, right? So if you take an SCA tool and you scan the official WordPress image on Docker Hub, how many of them can tell you WordPress is in the image? Almost not. I think we've come across one that can tell you. And why is that? Because the way it installs WordPress is it like clones the repo? It follows the instructions on how to build a release, and it copies that into the final image. Right? There's no metadata around telling us the tools and what went into the image? Um, if you look at our WordPress images, everything that goes into our images comes through a system package, and every one of those system packages leave a record. I said, Distroless, solve the bill of materials problem years ago, and then people, you know, started to add stuff on top of that because they didn't move fast enough. Well, that package database covers everything going into our images. So every one of our images will tell you exactly what version of WordPress is in there. And so I think in this case in particular, right. Like I think this the node vulnerabilities that we showed was in like Lib and TTP, which, you know, because we're building and packaging this software as part of our distribution, it shows up in the scan. I actually did a manual scan of node when I was digging into this. The node latest and node slim of the like almost a thousand CVEs in the node latest image. That wasn't actually even one of them. So there's stuff in there that is not finding. And this is just

one of the ways that we we work with SCA tools. We do things like turning on, um, you know, auditable mode in, in rust compilations and all kinds of other things to try and make SCA tools work exceptionally well with our images so that we can find out about the things and then fix the things, right. And so if you want to check out, um, the, you know, uh, the images that are available on how they might compare to, um, you know, what you have, uh, today, uh, I go to images and, um, you know, drop in the, uh, the name of the software that you're using. And I'm curious how it shows up and if we don't have it, um, you know, you can you can always request more. The growth of this catalog is entirely driven through customer requests, and we add several hundred images a quarter to this catalog. I think we've already added 400 this quarter and we've got a couple weeks left. So, um, so it is growing rapidly and that growth is entirely through customer requests these days.

Dev Tagare: Incredible.

Nancy Wang: Yeah, I love the visual representation too.

Dev Tagare: So so easy to draw.

Matt Moore: In our console. It also you can do that same view where you select the images. It shows you that same view over all of the images you're entitled to. And so you can see that same view across your entire organization. And a lot of our customers requested that, um, you know, because they had implemented Chainguard and, uh, you know, a year later they're coming up for renewal. And they were they were saying, well, I need a way of like quantifying the value we've gotten. And so there's trend lines of like accumulation of CVEs over time, what they didn't have to deal with and where they would be if they hadn't implemented Chainguard. Just being able to see that side by side of like these are the images I was using. These are the Chainguard images. And in aggregate, like, you know, it's a pretty, pretty steep contrast.

Nancy Wang: Thank you so much for that demo, Matt. I think this wraps up our episode. Um, Matt, sorry to keep you over. I'm sure you have, uh, some other meetings or customer meetings that you've we've made you late, too.

Matt Moore: Thanks for having me. Zero-Shot Learning is presented by 1Password.