Zero-Shot Learning - Episode 3: Travis McPeak from Cursor The new attack surface writes code

Hosts

- **Nancy Wang** - CTO, 1Password
- **Dev Tagare** - Sr. Director, Head of Engineering, Gemini Enterprise & Business, Google *(personal capacity)*

Guest

- **Travis McPeak** - Head of Security, Cursor

Opening

**Nancy Wang:** Hi everyone, I'm Nancy Wang, Chief Technology Officer at 1Password. Welcome to Zero-Shot Learning, which is a podcast about the reality of developing with AI from the people actually doing the work. I host this show alongside Dev Tagare, who's a Senior Director and Head of Engineering for Gemini Enterprise and Business at Google.

Today, we're going to be talking with Travis McPeak, a long-time friend of mine who's now at Cursor. Travis has spent his career securing real systems across cloud, application security, and product security at places like Netflix, Databricks, IBM, HP, and Symantec. All that experience has taught him that businesses must be made to move - so he's taken a realistic approach to security. Travis has also realized that teams will adopt new tools, and it is security's job to help them do so safely. In my opinion, Travis is one of those security leaders who brings both conviction and pragmatism. He knows where the risks are, but he also knows that being perfect kills progress.

So we're going to talk about what it means to secure an AI-native company like Cursor, how security teams should support the business without blocking it, and how to think clearly when the technology is moving faster than the guardrails you can put around it. Lock in. This is Zero-Shot Learning.

Episode 3

**Nancy Wang:** We have the pleasure of having Travis McPeak in our studio today.

**Travis McPeak:** Thanks for having me.

**Nancy Wang:** You've helped build security in some very different worlds - Netflix-scale cloud, Databricks, pretty massive product and platform security - and now AI-first coding workflows at Cursor. What's one hard constraint that never goes away, that you find people tend to underestimate every time the tech stack changes?

1. The Hardest Constraint in Security: Chaos

**Travis McPeak:** Yeah. So I think the hardest thing in security is always just chaos. There's a quote from a long time ago about the NSA's elite hacking group - Tailored Access Operations - where they would come into an organization and just be quiet and observe what's going on. And by the time they would go do whatever it is they had to do, they understood the inner workings of the business and complex systems better than anybody who worked there. And the reason this is so hard in security is because for a business to function as a good business, things are moving forward, things are changing all the time. There's a lot of work being done that you just have no visibility into. You often find out things late, and then on top of that, security is already spread thin. There's large stuff changing that may introduce risk. So security has to be very, very choosy with how and where they get involved. I think that's the number one dynamic - we're always going to have more that we have to be doing than we can actually do.

2. Securing Agents and the Identity Problem

**Nancy Wang:** So given that the volume is so much greater than what's possible to get done - let's talk through how that translates into your workload today. If you're securing an agent and not the user, how does that change things?

**Travis McPeak:** It's a really interesting and still unanswered question in the industry: identity. Like, what identity is this agent? Is it the identity of the user that launched it? Is it the identity of where it's running from - the platform, like an AWS instance? Does it have its own identity? I don't think anybody has really figured that out. And I assume, like most things in the industry, we're going to get completely different answers to that question. So first you have identity - as we've always had. And then you have to go and grant that identity access to things, which is IAM and access control - which we've also historically been awful at as an industry when it comes to setting things up with least privilege. I did a talk on this once - we've been talking about least privilege since 1970 and it's never been done well. Over-permissive things have always caused tons of problems.

At Netflix, I did a project called Repo Kid, which I think is like the best cut of this problem - you deliberately give a little bit more than you think anybody needs, then you observe over time and cut it down to the right size. And then of course when you do that and someone needs to add permissions, you need a system for doing that. I think that's kind of the best we can get. And what's interesting right now is that everything is moving so fast, driven by transformational technology that people really want to adopt into their business. And we in security just don't have great answers for what we're going to do here. At the same time, many businesses are not going to wait three years for the security team to figure it out. So we're acknowledging that businesses want to adopt this technology, we can't just throw up our arms and say "this is hard" - we have to do something. And so we're trying to adapt old practices and technology onto a vastly new thing, with varying degrees of success.

**Nancy Wang:** Well, I do love the identity shout-out, because that's one thing we're spending a lot of time thinking about here at 1Password - how to secure identities. Dev, what about you? I'm sure you're seeing that as well.

3. Secure by Default Without Slowing Down

**Dev Tagare:** Totally. I come from, like, the opposite world of Travis - it's mostly build, build, build, ship, ship, ship, all the time. So the top question in my mind is: what's your definition of "secure by default" that doesn't slow down teams? And where would you draw the line of "hey, this is the baseline, I don't care how long it takes, you have to get here"?

**Travis McPeak:** Like one thing I believe about security teams is they exist to serve the business. Your business exists for a reason. And security teams that obstruct that reason in a meaningful way are not good security teams. Your job is to do the best risk minimization you can within the amount of slowdown that a business will tolerate. Ideal solutions not only don't slow things down - they actually get faster. Like, you in security delivered a system that makes things secure by default, sets things up well, and it's actually faster and easier than the old system. Those are very, very rare, especially with the types of people that do security work. Net neutral is like: the processes just happen on top of it. You add security, it's passive, does absolutely nothing until the point when it saves you. And then of course many, many security teams are in "we're going to slow down the business as much as we can and still get away with it" kind of mode.

So yeah, I'm actually more in your philosophy of what you said. Like, I'm in ship, ship, ship too - that's what I want. I work here, I want the company to be very successful. There are obviously some very bad security things that it's my job to help us mitigate and buy down risk. But the biggest thing I want is for us to be successful and thrive as a business. So my usual

answer is not "how much can I block you?" It's "okay, yeah, I see why this thing you're doing is very important. How can I support you in getting that done in a way that's like what you wanted to do, but safer for this reason that I don't even want you to have to worry about?"

4. Everything as Code in the AI Age

**Dev Tagare:** Totally. And in this spectrum of options to support and accelerate the business from a security perspective - what do you think actually sticks? Is it permissions templates, the Everything as Code paradigm, reducing the entry barrier, better communication back and forth? What's been the most successful pattern or technique you've seen?

**Travis McPeak:** Well, Everything as Code has become surprisingly hot. Prior to joining Cursor in June, I was running a business called Resourcely - secure-by-default infrastructure patterns for developers. We worked our way into producing Terraform as the way we did that - we would produce Terraform patterns with standard templates, then merge in the users' opinions and security policies and deploy Terraform. Now what's become really interesting about the AI revolution is we have these wonderful tools that operate very well on anything that's text-based. So now if you can do configuration as code, that's very powerful because the LLM can produce it. And you can also have other people, or independent LLMs, come and review it. Once everything's good, it just gets auto-deployed. I find myself reaching for that even more than I did before.

5. Trust Boundaries in the Coding Pipeline

**Nancy Wang:** Sure. So moving into one of my favorite sections of the podcast - let's get deeper. As developers, we often talk about securing the pipeline, but it's really a lot of theory. So if you take a look at the coding workflow - very similar to how we used to write code before LLMs - you think about local to remote, pushing from your local machine to your repo, then from CI runners to the cloud. What are some trust boundaries that really matter in your opinion?

**Travis McPeak:** Yeah, I mean, everyone's got a different philosophy about this, and there are a lot of cultural considerations. For example, some companies with very good security teams take an extremely locked-down laptop approach - they might not even allow their developers to run with privileges on their own machine. My general approach is that's too obstructive to a business. So my north star is mostly just to treat the laptops as compromised at some rate, and then act accordingly. Anything coming from those laptops to anything else gets an increasing level of scrutiny from me. CI is one - if you're an engineer, that's probably the thing you're going to touch directly that gets scrutiny. And then CI gets bundled up and deployed on infra, and that gets scrutiny. The stuff that's happening on infra, changes to it, all of that gets a lot of scrutiny.

And then at the end of the day, any good security team - as I said at the beginning - has so much more to cover than we can actually cover. If you sat me down in a room and said, "for your life, you have to make a list of 50,000 really sketchy things that are going on," I think any security team can do that. There's just an infinite number of things that are not optimal, not up to standard. So really the art in this is being risk-based. For me, my number one crown jewel is customer data. And so all of those systems I talked about up until this point are a tier below wherever that customer data flows. I spend a ton of time on that - infrastructure hardening around it, auditing, all the usual good stuff.

6. The Capability Model for Agents

**Dev Tagare:** You know, I'm very curious about security-first approaches, which is actually why Nancy and I are friends. And I'm not a great security practitioner in that sense. So one of my key questions here is: there are many security approaches that assume the developer is a careful operator. Agentic tools sort of flip that model. In this case, systems can take actions at machine speed, autonomously. So in the new paradigm, what's your recommended capability model for agents - think scopes, approvals, break-glass, time-based, role-based? How do you think about the paradigm where agents are effectively just YOLOing it in the world?

**Travis McPeak:** Yeah, excited to answer that. I want to push back on a question to you really fast - what do you mean by security-first?

**Dev Tagare:** Yeah. So the thing is, when I go back and I author an agent, write a plugin, run it against the plugin - it could talk to third parties, it could inherit a bunch of permissions. In this paradigm what I'm thinking is that naturally giving super large permission sets is not a great idea. It's very viral these days, but doesn't seem like the most promising idea longer term, or even for enterprises. With that in perspective, I think there has to be a minimum barrier to make agents viable, to be shared, to be run in enterprises, and so on. So I'm really homing in on what are the table stakes to make these even usable in a shareable setting.

**Travis McPeak:** So you mean like security-first for agents, not security-first industry-wide? Okay. Yeah, like secure by default. In fact, that's actually the reason why we launched Cursor Hooks. That makes more sense, because my first thing was going to be challenging the "developers are super trusted" premise. I don't feel that way. I've seen so many studies about defect rates - we bring in all of these poor developers every year and make them do the full developer security training thing and it's annoying for them and they forget it because they don't use it all the time.

**Dev Tagare:** Oh, by the way - I began the question with "we assume that they are super trusted," but the implication was that they're not. Because we all forget our trainings.

**Travis McPeak:** Yeah. Well, and that's the reason that least privilege is important, right? Like if every developer was actually a security person first, we'd never get anything done - but secondly, you could give them more permissions and wouldn't have to worry so much about least privilege all the time. Anyway, we don't live in that world. So developers weren't mega trusted in the first place - not because they're bad people, they just have other priorities.

Now, we're talking about agents. There's this specter of prompt injection, which has been around since the beginning. I think it's hilarious that yet another asterisk injection has come and plagued the new thing - we just never got that separation of control plane and data plane right in the history of computers. Anyway, this is the new evolution of that.

And I still think people don't understand how non-deterministic these systems are. You write code - code has bugs, but modulo the bugs, it does that thing all the time. And then you get like very reliable systems - AWS IAM, for example, doesn't really have bugs for the most part. Agents aren't like that. It could be green, green, green, green, green, green - and then purple for no reason - phase of the moon, some inference bug, any number of things people don't understand.

So basically, the net of all of that is that you always have to treat an individual agent as untrusted - especially if it has access to something that really matters. That's why we have all these mechanisms in Cursor about you approving what the agent can do that might impact stuff you really care about.

And now we're all in this big mad rush - wouldn't it be awesome if the agent could just handle all the customer emails and then issue refunds? The question becomes, what about when it does something wrong? I was listening to Risky Business this morning - I heard some company accidentally gave away like 44 billion of bitcoin or something.

**Nancy Wang:** Yeah - actually, it wasn't an agent thing. They meant to give Korean won and picked the wrong currency.

**Dev Tagare:** Oh right. Yeah. They issued like 2,000 bitcoin or something crazy.

**Travis McPeak:** 20,000. It was some giant number. But now we're going to have that kind of stuff, right - where we set up the agent in a hurry because "it's going to be so cool, we don't have to be answering these customer requests all the time," and it's got the tool and we forgot to put in safeguards, guardrails, logging, anomaly detection, all of that. So I think the easy answer that we were talking about since the beginning of AI is: have one agent that's unaffected supervising the behavior of another agent. And then as the industry matures, we're going to get more normal anomaly detection kind of stuff. The agent says green 999 times out of 1000, and every time it says purple, we slow it down and look at all the input it consumed and its reasoning and how it got there - and maybe it can continue anyway. But that's the point where we slow it down.

7. Credentials and Identity in the Agentic Mesh

**Dev Tagare:** It makes a lot of sense. So taking this argument further, Travis - now imagine a world where the one-agent paradigm shifts to the agentic mesh, where thousands of developers are writing these agents. There could be a central orchestrator, but it's unlikely to govern all 1,000 agents - it's probably just facilitating "are you in my mesh or not," a kind of control-plane semantic. Now agents are talking to each other - could be agent-to-agent, could be tool call invocations via MCP, any new paradigm that might come up in this world. Do you think credential management becomes more persistent, or does it go down the ephemeral path? How would identity exchange and scope exchange happen in a true agentic mesh?

**Travis McPeak:** I mean, we're going to have a couple of iterations of this. It would be awesome if we had just-in-time ephemeral tokens used by an agent, with all of the AuthZ knobs we want to tune for it - we could do that granularly, based on profiles, it can request new ones - all of that. But that's a whole bunch of existing tech that has to change. So I think in the beginning we're going to have more coarse-grained controls here - like, do you want to offer it this tool or not? That could be a way of turning it on and off. And then beyond that, we've already seen tons of MCP brokers where access controls can already be done. That's one intermediate step we'll get to. But long term, I think what'll probably happen is we're going to rewrite significant parts of the AuthN and AuthZ stack here to make these things work better.

**Nancy Wang:** Do you think it starts to look more and more like: here's agent one, here's agent two, there's a gateway, and you do all of your policy enforcement at the gateway - and then it just looks like microservices again, except we call them agents now?

**Travis McPeak:** Yeah. I have a colleague, and at first when he said it I thought he was joking, but now I think he's actually right: everything in security ends up being a proxy. It's just proxies all the way down. So yeah, the concept of the agentic mesh does sketch me out a little bit, but yeah - swarms, galaxies, whatever - we keep using increasingly scary terms to describe this thing.

8. Using AI to Do Security Work

**Nancy Wang:** So we've been talking about securing agents. Now, one thing we're also hearing about - which I'm sure you've read about in Risky Business or other places - is actually using AI to do security. When Claude Opus came out, I think the whole security world was in for a surprise, because what researchers could do with red teaming and discovering new vulns was almost expert-level. And so when we think about this new landscape of static analysis tools like Bandit and SAST tooling - what still applies and what doesn't?

**Travis McPeak:** So I will admit - when GPT came out, I was caught off guard. I'd obviously been hearing about AI forever, and long predating me, vendors in security were selling AI snake oil that didn't end up being good. So I had this part of my brain that just learned to turn off - "yeah, it's snake oil, it's BS." And so I missed the actual AI takeoff until ChatGPT came out, and then I was like, wow, this is actually very interesting. And since then I've been following it like a hawk.

For me personally, Opus felt like a significantly better model - but not like "wow, this is a whole new thing." We'd had increasingly good models up to that point. So I've been using it for security work since around January of last year. It's definitely at the point now where you have to put less into it. People that were building at the app layer, doing all this juggling to just keep it on track and make it do stuff - a lot of that is less relevant than it had to be. But anyway, the way I would describe it is: earlier I said security teams just can't keep up with everything. There's way more stuff going on than we have time to deal with. Examples would be like careful code review - wouldn't it be awesome if everybody in the security org could in-depth review every single change? Sounds great - completely impractical. We're never going to be 1-to-1 with developers, and those security reviews, if you're being careful, take a long time.

But now when you have very sophisticated systems that you pay by token, that can just do it and never get bored and have full attention, we can lower the bar for what we can review. That would be a good example. Even just looking for the nature of change and getting the security team in at the right time - like, I care about X, Y, and Z. Look at every single change and your only job is to tell me if X, Y, and Z are happening. And if they are, flag me and bring me in.

Threat modeling didn't scale before - I did that for a whole year in 2015, it was awful, devs hated it. That kind of stuff actually scales now because you can have an agent go and produce an architecture diagram, go look for potential attack vectors, then go match that up to any code that was written and see if those are mitigated. I think that's a huge win. And I'm surprised that people are just now ramping into the idea that agents are going to be great for security - but better late than never.

9. AI-Generated Code: No Special Treatment Needed

**Nancy Wang:** And what about cases where you have AI generating code for you? Do you treat those diffs differently from a policy review or gating perspective?

**Travis McPeak:** No, not at all. There's this term "vibe coding" which everyone uses. I don't like that term at all for what I produce - to me, vibe coding is agent go do work, and it produces a bunch of code and I'm like "yeah, sounds good, merge it." That's not what I do. When I develop something, I obviously use AI because I'm so much faster and better with it. But I use it to ramp myself into new knowledge domains. If I'm doing something I don't understand how it works, I'll just question it, get an answer, I'm like "I still don't understand it," ask the question differently, get another answer, I'm like "I don't believe you, show me the docs." You just ask questions so fast. And if you want it line by line, you just copy and paste three lines into the AI and ask it to explain in as much detail as you need.

And then also - it or other agents will find bugs in the implementation that maybe I wouldn't have seen, or I would have seen but it'll find them earlier. So the net product of me writing something with AI is: one, I understand it much better than if I did it myself - because previously, I'm like "I don't quite get this thing, but it'll take me three hours to research, I'll just live with not fully understanding it." Now there's no friction - within two minutes I can have an answer with doc snippets to prove it's not a hallucination. It lowers the bar to my understanding. And there are multiple sets of eyes going and looking at it and finding problems. Anytime you want, you just spin an agent and say "go look for A, B, and C class of issue on this PR, tell me anything" - it'll go in, eagerly search, and then you can have another agent go and triage all of those, get rid of the ones that are not verifiable, and then you yourself look at and interrogate the remaining results.

10. Least Privilege for Tool-Using Agents in 2026

**Nancy Wang:** So Travis, rewinding a little bit - Repo Kid was about making least privilege achievable in a high-velocity environment like Netflix. What's the 2026 version of that idea for tool-using agents?

**Travis McPeak:** Well, I'll do the 2026 version for non-tool-using agents first. AWS IAM is a great technology - honestly it's probably the best implementation of this thing. But even me, I've dealt with it for years and it's like, "how does this condition work? What's the syntax like?" It's just so hard for anybody to understand. But models - their default training is better than 99.99% of people for that. So it's just going to produce better output than what you copied from Stack Overflow in the past. And then you could ask questions about it and get it even further improved. The security team can just go ask somebody else's agent "why did you pick that?" - and maybe it'll resolve itself. So the floor has gone up for anybody using AI in terms of how good that stuff's going to be. You can also just tell the agent: go look at all of the code and suggest the least-privileged set of permissions I need here.

Now, what's the agentic version? I think what we want is standing no permissions - or least permissions - and then based on the actual agent and its profile and what it's supposed to be able to do, there should be some requested, auditable set that it receives. You definitely want to know what it was granted and what it actually ended up using. And then there should be an escalation path. So if my agent's working on something and just gets hard-blocked at a point, it might crap out - that's bad for everybody using it. But if it's like, "you know what I actually need? The CEO's salary from the system" - super weird, never asked for this thing before - there should be a set of approvers that can grant it for this agent. To an extreme example, that kind of thing is what we're going to want.

11. Agent Trust Scores and the Thin Client Model

**Dev Tagare:** All right, that makes a lot of sense. And it talks to me a lot because we've been thinking about doing work on agent behaviors - almost like assigning a scorecard over time on how the agent is evolving.

**Travis McPeak:** Yeah. Like an employee, right.

**Dev Tagare:** Kind of like an employee - like an appraisal or perf review over time, but modeled on agent behavior or trust and safety scores. And to that, we've been thinking a lot about how to right-size the permissions or scopes for a given agent somewhat autonomously, without too much human intervention. Where do you think the equilibrium possibly could be - where you have some notion of a health score or a trust score - and how would you go about getting that signal in and assigning it to agents? Still agentic, or more deterministic?

**Travis McPeak:** Yeah, I mean, there's really cool stuff here. So you deploy an agent, that agent has a purpose, and you should be able to use some combination of patterns and other agents to analyze that and come up with an expected set of tools that it's going to be using - calls, skills, whatever. You have this baseline, and then that becomes a static list. Any request outside of that list could be: you don't have the permissions and you need to request them. Or it could fire your anomaly detection - there's a whole bunch of stuff you could do here because the baseline normal behavior of a well-formed agent... you don't want to have this megalith agent that does all the things at the company. You want purpose-driven agents. That way you can baseline them and give them the right access.

**Dev Tagare:** And on that paradigm - where do you think skills come into play? We're talking a lot about agent skills lately. They seem to be moving agents in a direction of one single omni-agent that has access to a bunch of tools depending on the role and scope, and then you mutate agent behaviors with skills. That maybe goes against the purpose-driven paradigm a little bit?

**Travis McPeak:** No, I don't think so. I think you just attach the baseline normal to the skills. So the main agent is like a hollow vessel that has some basic instructions about how to find and use skills at once - it's general purpose - but then you just apply the access control to the skills that are loaded. Makes sense.

**Dev Tagare:** Yeah. And I think it'll end up mirroring what we were talking about earlier - agents as thin clients. That's my thesis. I think agents are getting thinner and thinner over time, and that time is now.

**Travis McPeak:** Yeah. I mean, the MCP thing made sense to get something off the ground, but long term, loading every possible tool the agent might need into the context is not the way we're going to end up doing this.

**Dev Tagare:** Yeah, a lot of that functionality is also moving to the models themselves. Models are able to do deep retrievals, have file system access or a representation of a file system, and so on. So at that point, what was tool use from an MCP-style paradigm - if it ends up moving closer to the models - then the agents effectively become an orchestrator or a hollow vessel like you described, Travis. And all these primitive skills basically end up guiding you towards the outcome. So in a way, less autonomy - but more directed autonomy - which might end up making them more useful than they are today.

**Travis McPeak:** Well, yeah. If you can do the skills bundle, then you can say this workload gets these five skills - it can access those five skills, which means it gets the instructions about how to use them, it gets the credentials it needs to deal with those systems. And if it's not in your payload, you don't have those skills. And that's the request boundary.

**Dev Tagare:** And this is very similar - to your point, Travis - to how lazy loading was done. Exactly. So again, a lot of software patterns are repeating themselves, manifesting a bit differently with agents.

12. Open Claude and the Personal Assistant Problem

**Nancy Wang:** With Open Claude here - how would you think about scoping down permissions and making sure that it only took actions on certain data sources that you wanted it to?

**Travis McPeak:** What I might do is give my Open Claude a persona that's separate from mine. So it can send email as itself - it can't send an email as me. I might give it a tool where it has some data that I want it to process, and that's just-in-time - it automatically gets purged so that if something else later comes down and compromises it, it doesn't get all the data I'd ever had. I would just put it in as small of a box as possible for it to get its job done.

**Nancy Wang:** That's actually spawning conversations around secure execution or runtime environments. What do you think about the future of that versus everyone having to go buy their own Mac minis to run Open Claude or other agents?

**Travis McPeak:** The Mac mini thing is honestly really funny to watch. I don't understand why people want to do that at all. We have the basic stuff you would need to run Open Claude - it's called an AWS instance. We've had that forever. The people doing Mac mini want to run their own models and whatever. I think it's cute and fascinating that so many people are really into LLMs because of Open Claude, but that's not the way to go. There's so much technology that they're voluntarily bringing on themselves when they don't have to.

At the end of the day, I think what the whole Open Claude rise and craze speaks to is that people really want the personal assistant. And my take is that this is not an unfamiliar concept to Google or OpenAI or Anthropic. It's not like nobody at those companies had that idea. It's that producing that service and warranting it as a big company is risky - something is going to happen, you're going to get sued. So releasing it open source is genius because there's nobody to sue. You just go use it at your own risk.

**Nancy Wang:** I completely agree with that. The number of implications for a personal assistant acting on your behalf is amazing - and will likely end up being super litigious for any real company. Yeah, that makes a lot of sense.

13. Builder's Mindset: The Personal Trainer App

**Nancy Wang:** Travis, shifting gears a little bit - we call this section the Builder's Mindset. My key question to you is: if you had a month off to work on anything you wanted, what would it be?

**Travis McPeak:** Okay. So I had not a month off or anywhere close to that, but I did have about 50% of Christmas break as actual downtime. Some of that was golfing and mountain biking and stuff. But I had a personal trainer - a person who would program workouts for me over the internet. And then they told me they were getting out of it and not going to do it anymore. I was paying 120 bucks a month, so I was like, I bet I could actually do a really good job of this myself.

And I built a GPT loop with tools hosted in Lambda and DynamoDB, fronted by an API gateway. And then I vibe-coded - legit vibe-coded - I have never written iPhone code in anything I've produced, but I've now coded iPhone and Apple Watch apps. I've still never touched Swift. I just asked the LLM to do it anyway. It's great. I've been using it for a month and a half, and it's better than what I had before - and I don't pay anywhere close to 120 a month. Like five dollars of OpenAI credits and very minimal AWS hosting. That's my builder's mindset.

**Nancy Wang:** I built one of those. Incredible - truly a builder's mindset.

14. What Changes in Security in the Next 6-12 Months

**Nancy Wang:** Well, let's bring it home with a final AI security question. A lot of teams these days are still talking about AI security workflows like it's 2019. What do you think is the biggest thing that's going to change in the next, let's say, 6 to 12 months?

**Travis McPeak:** I used to say years - but honestly, it's months these days. So there are people who embrace security engineering - their job is to use engineering principles and go integrate with where the security risk is. And then there are security governance people - the camp whose job is not to go solve anything, but to buy a bunch of tools, make sure the right people who might benefit from findings get them, and then chase them down and make sure they hit things within the SLA.

My biggest hope is that that second group is just going to retire. AI is going to be so good at chasing tickets that if you can't actually build stuff and provide value to the business, you go find something else to do for work. We have so many people in security that have little or no value - or sometimes are destructive to the business they operate in. I hope they go away.

And then the people who are left and do security engineering - security people are usually very good at thinking about systems, asking hard questions, wondering why, reasoning down to first principles. They're now going to be able to touch a lot of systems they couldn't before because their ramp-in time is so fast. This is what I've gone through personally. Like, I was never the fastest programmer, but AI was the perfect partner for me - it can handle a lot of the code, I can go interrogate the code and the design decisions, and we converge on something good. Or if I need to go touch a system I haven't touched before, I can learn about it in an hour. It would have taken me days before. So it just lowers my bar for where I can touch and have impact.

And so if we wanted to, as an industry, get better security coverage than we had before, we could do that with fewer people. And if my hope comes true and all these ticket-pushers retire and start doing something else - then those who want to do security engineering, there will be more of us. We'll go diffuse to all of the businesses that haven't been able to hire talent like that before, and overall security will improve. That's like my utopia vision for this.

**Nancy Wang:** I love the hot takes, Travis. Love the hot takes. All right - thank you so much for joining us. We had a great time. I hope you did as well. And stay in touch.

**Travis McPeak:** It's fun. Thank you. We'd love to hear from your engineering teams.

*Zero-Shot Learning is presented by 1Password.*