Transcript: Zero Shot Learning Episode 1: Fotis Chantzis from OpenAI

English (US)

00:00:01.960 — 00:00:15.360 · Speaker 1
Hey, everybody. Welcome to the Zero-Shot Learning podcast series. In this series, we focus on talking with AI builders who are actually doing the work. My name is Nancy Wang and I'm the chief technology officer at One Password.

00:00:15.520 — 00:00:28.280 · Speaker 2
And I'm Dev Tagare. I'm a senior director and head of engineering for Gemini Enterprise and Gemini Business at Google. But for lawyers listening, the views are my personal views and have nothing to do with Google or Gemini.

00:00:28.400 — 00:00:46.600 · Speaker 1
And in AI, why we've named our podcast Zero-Shot Learning is Zero-shot learning is the idea that a system can take what it already knows and apply it to new situations. And so that's exactly what we do here on this series, which is talking to AI builders, applying what they know to new problems and new challenges.

00:00:46.640 — 00:01:03.650 · Speaker 2
Each episode we go into the workflows and development shaping the tools we use in our day to day lives, as well as on enterprise settings. And Nancy and I really like to get into the weeds of AI, where the puck is moving, what the different acronyms are and how builders are building.

00:01:03.690 — 00:01:19.569 · Speaker 1
But also what that said, uh, do stick with us because whenever the conversation gets too technical, you know, maybe we'll sometimes catch ourselves and define some acronyms. Maybe ask the builders who are also on our episodes, to go a little bit deeper into what it actually means, aside from the jargon

00:01:19.570 — 00:01:20.129 · Speaker 2
In our first

00:01:20.130 — 00:01:20.530 · Speaker 1
episode,

00:01:20.530 — 00:01:24.129 · Speaker 2
we spoke with Fotis[Chantzis], he's an agent security leader at

00:01:24.130 — 00:01:24.810 · Speaker 1
OpenAI,

00:01:25.130 — 00:01:27.610 · Speaker 2
about one of the biggest unsolved problems in

00:01:27.610 — 00:01:28.010 · Speaker 1
AI,

00:01:28.050 — 00:01:28.289 · Speaker 2
which

00:01:28.290 — 00:01:28.930 · Speaker 1
is,

00:01:29.090 — 00:01:29.210 · Speaker 2
you

00:01:29.210 — 00:01:29.410 · Speaker 1
know,

00:01:29.450 — 00:01:31.410 · Speaker 2
identity access and controls for

00:01:31.410 — 00:01:31.770 · Speaker 1
agent.

00:01:31.810 — 00:01:35.330 · Speaker 2
Everyone agrees that the biggest next door in AI is going to be led by

00:01:35.330 — 00:01:35.930 · Speaker 1
agents.

00:01:35.930 — 00:01:38.170 · Speaker 2
They need their own authentication

00:01:38.170 — 00:01:38.930 · Speaker 1
mechanisms.

00:01:38.810 — 00:01:52.570 · Speaker 2
They need their own personas and identity management systems. Because there's a real security risk, if they were accessing some data that they're not supposed to or doing actions that they are not authorized to do.

00:01:52.610 — 00:02:10.190 · Speaker 1
Yeah. So, for example, let's consider an example where you have an agent summarizing your emails, or maybe even summarizing your sensitive slack DMs and sending you an email notification. Well, in that case, which slacks? Which channels is the agent actually allowed to see? And who is it allowed to email on your behalf?

00:02:10.229 — 00:02:25.710 · Speaker 2
I think everyone is excited about agents, but once you start seeing the potential for security breaches, you cannot unsee them and you're just going to move away from agents. It's important that so many people are working on IAM at such a high level and with intensity right now.

00:02:25.750 — 00:02:36.870 · Speaker 1
Yeah, I mean us at one password include it as well. And certainly, you know, one of my favorite

aspects about this podcast series is in every episode, we learn a ton from the guests that we have on our show.

00:02:36.910 — 00:02:43.590 · Speaker 2
We've recorded a handful of episodes already for the season, but unfortunately I had to miss this specific one.

00:02:43.630 — 00:03:03.960 · Speaker 1
Well, you're certainly missed Dev. And, you know, we we had actually our VP of AI, Jeff Malnick, stand in for you on this particular episode, but certainly, you know, tune in for the rest of the series where Dev and I explored these topics with our AI builders, and some of them actually bring demos live demos to the studio.

00:03:04.200 — 00:03:08.880 · Speaker 2
Well, thank you, Jeff, for stepping in. And yeah, I'm excited about the rest of the demos.

00:03:20.240 — 00:03:25.800 · Speaker 1
Welcome to the Zero-Shot Learning podcast. I'm Nancy Wang and I'm the CTO at One Password.

00:03:25.840 — 00:03:32.240 · Speaker 3
My name is Jeff Malnick. I'm the, uh, general manager, VP, uh, head of developer and AI here at One Password.

00:03:31.920 — 00:03:38.120 · Speaker 1
And so today we're super excited to welcome Fotis Chantzis to our virtual studio here.

00:03:38.160 — 00:03:40.800 · Speaker 4
Thank you for having me. Um, this is this is great.

00:03:40.840 — 00:03:48.400 · Speaker 1
When we talk about the fact that agents need identity, what's the unit that you actually want identified for agents?

00:03:48.600 — 00:04:30.300 · Speaker 4
So, in my view, agent identities, uh, well, I non-human identity. Um, that can be directly authorized by a tenant or delegated by some kind of user to access local or remote resources to perform some kind of task. Um, then the unit, I guess, uh, can be the actor at the action boundary. So the thing that causes I like to, to kind of frame it as the some kind of effects So effects, uh, we mean, like any kind of tool calls, any kind of like, uh, API, you know. Right. Actions, payments, deployments, uh, and all of that. So, I mean, basically, I see the, uh, the

00:04:31.740 — 00:04:49.858 · Speaker 4
unit as like a bundle of, uh, of facets that kind of tie together, like, who run this, what code did run, where it run for, for what purpose and what session carried the action. So, yeah, that's how I think about it. But also would like to, uh. Yeah. To, to

00:04:51.620 — 00:04:53.030 · Speaker 4
get your thoughts on this?

00:04:53.150 — 00:04:54.510 · Speaker 1
Now, what about you, Jeff?

00:04:54.590 — 00:06:10.520 · Speaker 3
Yeah, I agree with everything Fotis just mentioned. Uh, what's interesting about agents is that they behave like humans, but they talk like machines. And so I think the boundary that Fotis pointed out here is, is an applicable one. Um, the one of the things I think when it comes to agent identity that's kind of challenging is the fact that, like a lot of the protocols and frameworks that we have today, they're super focused on federation and some of the things that we've implemented, like, you know, it's a natural to reach for OAuth, for example, for MCP, it's a very common framework, right? It's the one that's most well adopted. But you know what we what we oftentimes lack in some of those frameworks is the ability to set a real concrete identity. You need to layer onto OIDC on top of OAuth to get that sort of, um, that sort of data. And, we're now just starting to see people kind of reach for those frameworks to, to layer on and get a little bit more robust identity information that's grounded in something. In that case, you know, an identity provider. But yeah, I think that that's the you know what I'm hoping that I'm really looking forward to talking about more today is some of the nuances there. And you know, how we can construct a more durable agent identity framework?

00:06:11.760 — 00:06:27.800 · Speaker 1
Yeah. I mean, Jeff, you mentioned OAuth, right? OAuth was really integral for solving consent for deterministic apps. But maybe let's shift the conversation to talking about how agents are no

longer like traditional apps. And you know what happens when you think about an agent like a traditional client app.

00:06:28.160 — 00:08:18.390 · Speaker 3
Yeah. So I mean, the difference between an agent and a typical like client server architecture is you have all the same machinery, right? You have client server architecture, you have the this deterministic chassis, so to speak. The difference is that the control loops are driven by outputs from a non-deterministic source That's an inference source. Right. And that's what makes them pretty challenging. A lot of times, you know, what we're seeing out there is agents get spun up, uh, either on a laptop or a third party environment. And the way that we're delegating credentials or access to them is through a 2 or 3 legged OAuth flow. And the outcome of that is that whatever resource server you're accessing and that resource is basically protected by an authorization server. That authorization server, uh, is different every time you go to a different resource server. And so the subject that you're actually writing into that, uh, JWT that's admitted in that process through that delegation process changes. if you're a security team and you're trying to track what a specific agent, let's say on a laptop or a third party service, maybe it's running over a sandbox or up on, you know, using open AI.

Uh, you end up having different subject identifiers in all of these systems. It becomes really, really challenging. And so what we're you know, what I'm hoping is that we can, you know, start to align on some more consistent frameworks for applying each identity. The another interesting outcome is that the human that delegated the agent, um, isn't always clear. There are some emerging standards that are trying to, apply more, I would say durable, construct for identifying those humans, across agent actions. But we're not quite there yet in terms of an industry. But, Fotis what do you think?

00:08:18.590 — 00:08:25.909 · Speaker 4
Yeah. I mean, all of this resonates. The main problem here is that right now, I feel like

00:08:27.030 — 00:09:50.380 · Speaker 4
governance and attribution are something that, um, you know, fail easily, uh, without this framework. There is also, you know, all of these protocols initially, like OAuth and OIDC were built for stable, you know, apps that don't really change the... There is no no concept of continuous authorization that agents require because an agent tasks like basically it starts with one task and then decides that it needs to do something else.

So an example that, you know, we've seen, let's say you have Codex and you, uh, you assign it like a, some kind of, you know, write a feature about this that uses this library or like the agent decides to, um, then download this, uh, this library from, from, uh, from the internet, but then it it needs to, um, it needs to look up the API documentation of how to use this.

And then it's like the initial task that the user intended was just a coding task. But then throughout the, you know, the course of this task, the agent decided that, um, you know what? I actually need to access the web to, uh, to look up more documentation. So, if the user had initially intended to just, you know, limit it to, to

00:09:51.700 — 00:10:39.460 · Speaker 4
just local file operations for this coding thing that that changes. And then, if you, if you don't have like either some kind of like feedback loop to the, to the user, which, you know, sometimes can be annoying because you have decision fatigue and some like if you overdo it the user says, okay, approve, approve approve or like I don't care. And then that they run it in YOLO mode and then bad things can happen. Uh, and so that is I think that is the, the, the root, I guess, cause of the, of this problem that we are describing that, um, things are dynamic. And when OAuth and OIDC were built, um, they were meant to be used by humans, not not the dynamic workflow.

00:10:39.700 — 00:12:14.850 · Speaker 3
I want to add on one thing there, in terms of when you were talking about the the policy systems and how they're kind of lacking today. Um, agents behave like humans. They talk like machines. They also work in machine speed. And so all of these systems that we have in place today to do authorization are really designed around a level of load on those systems That's like one time flows, right? So you go through a three legged OAuth flow, you authorize something on your behalf that one time, and then that JWT is good for a bit. You know, you're not coming back right away to reassess policy or do maybe inference on what this thing did and then determine if that policy should still apply. These are what we're what we're describing here that we need to build is like this extremely high performance policy, an authorization system. That can do, you know, every lesson, a second policy, uh, authorizations, uh, for an agent action. And there's a lot of context and a lot of things that happen within a couple of seconds at machine speed and reassessing that continuously like that. don't think that there's really a policy system on the planet that can really handle the kind of load that we're going to see.

If, you know, an enterprise today spun up a few hundred agents per employee, for example. Right. That's like tens of thousands of agents, um, all doing that over and over again. So that's a really exciting problem for for me at least, I think that that's a that's a really cool distributed systems and security problem kind of built into one thing.

00:12:15.530 — 00:12:30.490 · Speaker 1
Yeah. So both of you have actually now talked about kind of the limitations of OAuth and existing protocols. So I'm just going to ask the hard question of the room, which is um, what's your answer to should agents be treated as first class principles with their own identity.

00:12:30.690 — 00:12:32.930 · Speaker 5
Yeah. So I mean, I think this is what.

00:12:32.970 — 00:12:33.969 · Speaker 4
OIDC

00:12:33.770 — 00:12:34.890 · Speaker 5
is trying to do.

00:12:35.090 — 00:12:36.330 · Speaker 4
Which I think is

00:12:37.370 — 00:12:41.700 · Speaker 4
is exciting. Uh, it doesn't completely solve the problem. Um.

00:12:42.820 — 00:12:43.260 · Speaker 5
I mean.

00:12:43.500 — 00:14:11.600 · Speaker 4
Maybe there's some confusion of the terms, though, because there is a there's the concept of like the the agent being its own service principle, uh, versus like acting on behalf of someone. And this these are two different things. And I think the right way to do this is that in most cases, you should have a human delegate to, uh, to, to an agent to do something.

So the the act on the, on behalf of kind of pattern. Um, and in the case of like when, when an agent has its own service principle, which honestly shouldn't be the majority of cases, you you still need to be able to attribute it to a human that at least instantiated the agent. This can be like an enterprise or something So, um, an example is that you have, you know, some kind of, um, uh, SEV bot that, uh, people are using in your company to kind of ask about an incident. And that kind of agent, uh, it wasn't necessarily deployed by, um, uh, by any, any engineer that is using the SEV bot as a sort of like interface to ask questions about an incident and all that. it was instantiated by, you know, I don't know, the IT admin of the company or like, there's still some kind of, uh, uh, you know, tracing it back to the human.

If we don't have that, then we lose attribution. We we lose governance. And then again, bad

things can happen.

00:14:11.600 — 00:14:44.210 · Speaker 1
And well, talking about like, authority, right. And attribution, um, maybe, like, you know, Jeff would love to have you chime in when we think about, like, sensitive data access. Right? So coming from the data world, that's kind of what I'm thinking about, which is now agents are accessing file systems.

Right. Um, local, you know, stuff that's stored on local disk. Um, how do you think about, for example, when enterprises need to go back, especially for, for auditing purposes and say, you know, which agent actually accessed this data set? And are you thinking like cryptographic proofs or just correlation?

00:14:44.330 — 00:16:06.460 · Speaker 3
Yeah, I think that's a really interesting follow on to Fotis bringing up OIDCA. One of the interesting things about OIDCA is it's a couple of new fields. Some of those new fields are related attestations. And so at the end of the day, what this is about is having a client prove what it is and where it's running. Right. And that gives you a you can create a cryptographic proof that this thing is actually what it is, and also that it's running in, say, an AWS Nitro Enclave where you could have some sensitive information that could operate on in a secure way. And you can actually bring that attestation, that cryptographic proof that's in that enclave, as part of the JWT. Um, and in terms of authorization of that thing. Right. The those JWT also come with fields like delegate or subject. So you can actually map a specific agent identifier to the human that actually spawned it. Um, and so you can do all sorts of sophisticated things now. Right? So you can say, can this agent actually run in this enclave? Um, is it allowed to actually grab this data set in the first place? And you can bind that privilege for that agent to what the human has access to in a really non reputable way, because it's bound to cryptographic proof. And so in terms of, you know, a lot of people are on the defense when it comes to agents. Right now we have a lot of DLP products coming out and those are necessary and relevant.

00:16:06.500 — 00:16:10.260 · Speaker 1
Don't forget CASB like solutions too - let's just firewall everything.

00:16:10.300 — 00:16:55.560 · Speaker 3
Exactly right. Um, MCP gateways everywhere. Um, and so I think that the, you know, uh, this is a more offensive scheme, right? This is about having the agent come in with some cryptographic proofs about what it can and cannot access. And, you know, I think there's a lot of really interesting, uh, stuff that's happening around confidential compute in the space right now.

Exactly. related to this, and I think that it blends really well with some of the stuff that is being proposed in OIDCA, and whether or not OIDCA is the final outcome of this. I think that it's a it's a meaningful step in the right direction to take the protocols that we do have today in the limited nature that they are, and get us to a point where we can actually do some more meaningful things around security for agents.

00:16:55.920 — 00:17:33.290 · Speaker 1
Yeah. And so going back to OIDCA, okay, I forgot, as you mentioned, this, and this brings us actually to this concept of delegation chains and how we should think about them. Right. So what is the scope what kind of systems you want those agents to target like their duration. Right. Different thresholds. I mean you also talked about controls, right?

Controls for agents when we started this conversation. And so, you know, given what Jeff just said, um, can you maybe walk us through like how you at OpenAI are thinking about delegation chains and whether you think they should be like a cross provider standard, or do you actually foresee every vendor inventing their own envelope at some point.

00:17:33.570 — 00:18:48.670 · Speaker 4
Yeah. I mean, ideally we don't, uh, have to to to reinvent the wheel, uh, from other. Every company does their own thing, and we, we try to standardize this. And we all agree that, you know, this is kind of the, the, the audit envelope and the, um, the delegation chain envelope that we can use. Um, I will say that, uh, this is going to be one of the most important things to solve.

By the way, how do we make, um, uh, basically task bound authorization for for agents. It can also be action bound. Like in a, I guess more specifically or obstruction. But I think tasks make sense to start with because they are, uh, a little bit more coarse and a little bit more manageable. Um, the I guess there's and there's no silver bullet, by the way.

So it really depends on, on where, uh, on the environment in which the agent runs. And in some cases, maybe you have even long standing permissions in other when the environment is such that you know its sandbox and maybe it has very little, uh, network access, and the data that the agent has access to is like not sensitive.

Now for I think the primary problem with all this kind of, you know,

00:18:49.790 — 00:20:58.050 · Speaker 4
creating delegation chains is that still, even if you even if you do that, which is important, it is the user intent is very hard to encode deterministically because natural language itself is ambiguous. So when the user says like, oh, I want to summarize my email, what does that actually mean? Does it mean that the agent is also allowed to to send summaries to Slack, or

writing these summaries to like a database or like create a linear ticket alongside it because the agent thought it was a good idea? Um, in, you know, we also have the problem then of not just prompt injection, which is, I think, you know, ties in very well with this discussion, which is that, um, the more sources the, uh, the agent has access to, the more attack surfaces, uh, it opens up.

And all of these, uh, sources are becoming, uh, prompt injection attack vectors. And in these cases, um, there's also other cases where you you might have this, you know, a misaligned agent or just an agent that thinks that, you know, it's trying to be helpful. And then it does this opportunistic like tool invocation, which is that.

Yeah. I thought, why not? I'll also like, uh, creates a slack message with your, uh, with your email summary because I thought it was a good idea, but then the emails contain something sensitive and then, um, there's, you know, uh, bad things happen. Um, and in going back to the dynamic, I guess, nature of these flows, uh, another problem there is that, um, you might have intent drift.

So, um, but going back to the example in that the, the, the original example with the coding agent where it decides to browse the web to find out about how a certain library works, although you just asked for for a new feature, all of these problems still, um, are still applicable. Um, even when you have this, you know, sort of, uh, delegation chain because again, of the, the nature of, uh, natural language, uh, intent.

00:20:58.610 — 00:21:10.210 · Speaker 1
Yeah. Zeroing in on intent, maybe. Jeff, uh, I'd love to get your thoughts because do you think, like the solve here is policy based consent some sort of, like, pre-approved capability grants or something else?

00:21:10.770 — 00:25:32.040 · Speaker 3
Wow. That's again taking a second to unpack. So let me, um. Yeah, let me let me position it this way. I want to start actually with maybe a little bit of history or actually where some of these challenges are actually deriving from. Um, so today, a lot of the problems that we have is because an agent can't authenticate.

And so when the agent does authenticate, it's authenticating into this system where the identity, the reason that it's authentication is valid is that you're trusting that a private key in an IDP somewhere is valid, and you trust that, right. And you trust it because of DNS most of the time, which is also a federated identity system at the end of the day And we've all seen how, um, spectacularly DNS can break when those trust chains get, you know, nefariously, uh, broken. And so I think that, you know, that is a it's a very brittle thing, this federated identity. Right. And the way that, um, you know, going back to the authentication story, the way that we've solved

workload identity in the past is by applying. And I promise I'm going to get to the intent side of this in second. But the way that we've solved the federated identity problem or the workload identity problem in the past is that we've basically relied on testing these controlled compute environments like Kubernetes and EC2, right? So if you want to allow an agent to automatically authenticate somewhere, you would test the environment that it's running in. And based on that attestation, you apply policy. Because you can be pretty sure that the agent or the workload that's spinning up in this Kubernetes cluster is going to behave in a very specific way. We can't make that assumption about agents, right. So testing that identity becomes really challenging. You can probably see the terrain and controlled compute environment, but one you don't know. As Fotis pointed out, we're kind of prompt injection might take place there. Um, and so you can't rely on a point in time, intent based policy application. Um, the second thing is agents aren't just spinning up on EC2 and Kubernetes anymore. They're on your laptop, they're on Vercel. Sandboxes. They're up in, uh, you know, in OpenAI's platform. They're everywhere on third party platforms. And so those environments are in uncontrolled. They're very, very challenging to attest. Um, we I spend some time trying to figure out how to prove the laptop wasn't lying to me. It's a really hard problem. Um, and so I think that the, you know, that to me says, okay, the federated identity system that we have is super brittle. Um, so applying policies based on federated identity is kind of weak, right?

Because you have all these chains of federation that could break, and therefore the policy that you apply might not be valid. Um, so what are the options in that world? Um, there are some formats out there. We're issuing models in the United States and in Europe these days that relies on a, um, it's a specification It's an ISO specification, one 8013 five based on the W3C standard for Verifiable Credentials. The cool thing about a VTC workflow is that the issuer, right, you have a centralized issuer, but it's not a federated issuer, right? You have a you have claims signed in that document by that issuer. And the the credential itself is bound by the private key that the holder has. And so every identity in the system has its own private key.

You're not relying on some centralized system and hoping that the chains of trust in between haven't been undermined. You know for sure that the credential that it's presenting is based on that private key. The cool thing about this system is that issuers along the way, and other holders along the way, can actually sign claims inside of that document that says what this entity is allowed to do In the case of mobile driver's license, the thing that you're allowed to do is prove that you're over 21 without presenting any other information through zero-knowledge proof, you can say you're over 21. Nobody has to know what your birthday is, what your address is, etc. those zero-knowledge proofs are super powerful. Um, but one of the cool things is that you're actually basically creating this very portable way of bringing your policy with the identity.

Um, and in that way, in a world where we're looking at how do we apply policies rapidly against changing intent. This is an area that I think has kind of been overlooked right now, but one that I'm sort of looking at is a possible avenue for solving some of those performance guarantees that we need around policy and based application.

00:25:32.080 — 00:25:58.880 · Speaker 1
Yeah. So I mean, even as someone who has worked in infra for for almost two decades now, right, that's that's a lot to chew on guys. Right. And you know, for all the the CTOs and CIOs who've been watching, you know, this episode may have photos back to you, I'm sure you get this question a lot. Like what's the biggest barrier, right.

When we think about things like authentication and authorization at very fundamental levels, like we're discussing now, are barriers to agents running in production.

00:25:59.200 — 00:28:01.070 · Speaker 4
I mean, as a it's a fundamentally a matter of trust. Um, and so trust boils down, um, well also to, to to identity and all of these things that we are discussing. Um, so, I mean, yes, you can have I think if I had to choose between, um, basically the task bound authorization versus the, uh, the, the attestation, I would say that, um, attestation maybe is is seems a little bit secondary in, in this sense because, um, there is there is still a big problem on, on how um,we, we are trying to like a lot of the enterprises are trying to, you know, to the, the most convenient thing that allows agents to run right now. But then, um, especially if we're talking about, um, you know, financially, uh, regulated institutions or other, uh, regulated industries, they are they are not going to trust these agents, uh, with accessing, um, very sensitive workflows. Um, so there is there's very little, um, there's no standard ways to, to do like policy, even if with all these standards, I think IDCA and, um, even Agentic JWTs are trying to to solve this, these problems that the older protocols didn't really support for agents But you still need to have a good, uh, policy layer, a policy engine. And, um, what we've seen is that, um, and most of the I think there was like some kind of, uh, a Gartner report that was saying that, you know, 80% or 85% of the companies actually don't even bother with writing very complicated policy rules. And I don't expect that to, to, to be the case for agents when you, you have something that is even more complicated because it's, you know, again, the there's intent drift and there's, uh, this, this dynamic workflows., um,

00:28:02.110 — 00:28:27.879 · Speaker 4
that is and then there are companies that want to you to, to leverage these, um, these, these capabilities. But, um, an example of what we did, which doesn't really. Doesn't even need a genetic identity. Which is kind of on the layer of security controls to to touch on that subject a little bit. Is the concept the notion of parameter constraints?

Uh, we published those, um, in the

00:28:28.920 — 00:29:13.940 · Speaker 4
early. Uh, before Christmas, uh, when we, uh, shipped our app ecosystem. And the concept is

very simple. You have, for example, an agent that connects to your Gmail and your your Google Drive. But obviously all of these now are becoming, you know, attack vectors for prompt injection. So what you can do instead as a, as an enterprise is to reduce the the attack surface by saying, uh, the agent, yes, they can access Gmail, but they will only trust or read and write email when that comes from a trusted sender.

So maybe you can limit the, uh, the parameters to just your corporate domain, assuming that, of course, like, you know, you trust your employees more than, you know, an external, um, you know.

00:29:13.980 — 00:29:14.420 · Speaker 3
OtherThat

00:29:14.420 — 00:29:16.540 · Speaker 1
will Yeah. But,

00:29:16.580 — 00:29:16.699 · Speaker 4
I

00:29:16.700 — 00:29:16.940 · Speaker 1
mean,

00:29:16.980 — 00:29:17.860 · Speaker 4
it's it's

00:29:18.020 — 00:29:19.500 · Speaker 1
fundamentally, uh,

00:29:19.500 — 00:29:20.499 · Speaker 4
reduces the

00:29:20.500 — 00:29:20.780 · Speaker 1
risk.

00:29:20.780 — 00:29:23.140 · Speaker 4
And then with Google Drive or like Dropbox or file

00:29:23.140 — 00:29:23.620 · Speaker 1
connectors,

00:29:23.620 — 00:29:25.459 · Speaker 4
you can tag files based

00:29:25.460 — 00:29:25.620 · Speaker 1
on,

00:29:25.620 — 00:29:25.820 · Speaker 4
like

00:29:25.820 — 00:29:26.220 · Speaker 1
the, um,

00:29:26.220 — 00:29:26.420 · Speaker 4
the

00:29:27.700 — 00:29:30.339 · Speaker 4
ones that you know or originate from a trusted

00:29:30.340 — 00:29:32.260 · Speaker 1
source. Again, like, um,

00:29:32.540 — 00:29:35.660 · Speaker 4
a very common attack that we've seen out in the world is

00:29:35.660 — 00:29:35.940 · Speaker 1
like,

00:29:35.980 — 00:29:37.740 · Speaker 4
you have some kind of attacker that

00:29:37.740 — 00:29:39.140 · Speaker 1
shares, um,

00:29:39.180 — 00:29:39.819 · Speaker 4
a random

00:29:39.820 — 00:29:41.060 · Speaker 1
file, uh,

00:29:41.180 — 00:29:41.699 · Speaker 4
with

00:29:41.700 — 00:29:42.260 · Speaker 1
you.

00:29:42.420 — 00:30:05.500 · Speaker 4
Uh, and then if you have an agent that automatically, like, ingests these files from Google Drive, then that file could be, you know, uh, contain some prompt detection payload and then that, uh, misdirect the agent to do something malicious. So, but the parameter constraints is, is a simple solution that, um, is basically reduces this, uh, this kind of risk.

00:30:05.540 — 00:30:27.710 · Speaker 1
And so, um, you know, maybe to summarize, right, for let's say if you're a CTO and you've been, you know, given the directive from the board, hey, you need to deploy internal agents, right, into prod. What are the first five controls that you would stipulate that they have? Like basically like a no excuses minimum for identity auth, you know, secrets handling and audit.

00:30:27.910 — 00:30:37.470 · Speaker 4
Yeah. So I mean you you definitely want short lived, uh, workload credentials. So ideally task bound to the, uh,

00:30:39.070 — 00:33:50.554 · Speaker 4
task bound instead of broad permissions. Um, you need some, some level of, uh, sandboxing as well for the agent. That also depends on the I think this is very, uh, context dependent. So if you have again, if you if we're talking about, uh, a cluster that is network isolated, then um, maybe it's, it's fine to, to, to have an agent be less um, to have kind of less of a human in the loop in those cases.

Um, but when you have Codex or like a coding agent or something running on your laptop, then for sensitive operations,you, you sometimes need to to ask the human. You definitely need some kind of credential management. So this is another interesting topic. Uh, we've we've also done a lot of work there, which is how do you prevent - I think it's it's essentially game over if, if, if a credential, any kind of secret material ends up in the context window of the agent, uh, because then this means that any kind of prompt injection and sort of misdirection then can lead to that credential being exfiltrated to an attacker controlled endpoint. so what you do instead is you either have this kind of, um, uh, connector gateway concept where you, you basically steer the agent or force it through some kind of network proxy layer to connect to these, uh, third party APIs through this connector gateway, where then it injects the the right credentials And so the credential, the gateway is outside the, the sandbox and outside the premises of the where the, the agent runtime is. So it's a, it's a controlled um infrastructure and it's trusted. Uh, that is one, one way to do this. The other way is to have some kind of, you know, authentication translation layer that could be again, um, for, uh, less structured, um, I guess when, for example, you don't have a connector implemented and you have, let's say, a developer that is building, you know, a payments app, and they need to to test with a stripe API key, and they want the agent to have access to that Stripe API key. What you can do instead is that you have a network proxy that, again, should sit outside the, uh, the premises of the agent, and then you have this, uh, credential injection thing where the you don't give the actual Stripe API key to the agent, you give it like some kind of placeholder credential, but then your network proxy when it sees this, then the request that goes to the, uh, to the stripe API endpoint API.Stripe.com or something, then it knows to inject that the the real credential there. then you basically bind the credential to the specific endpoint. So in a worst case scenario where uh the there's a the agent is prompt injected. Um, the worst case scenario is that, you know, it tries to uh, it will only be able to send that secret material to Stripe and not an attacker controlled endpoint. So that is, um, uh, the credential one is

00:33:53.670 — 00:34:26.090 · Speaker 4
a big, uh, control. And then I would say some, some kind of minimal audit schema. So because auditing is, is and attribution is super important. Uh, that also depends on your infrastructure, but you should be able to attribute, um, any kind of, you know, potential damage, uh, to, uh, back to an agent and then back to a human that instantiated this agent, or ask the agent to do something on their behalf. Uh, and of course, the policy enforcement that we talked about.

So I think these are, these are very fundamental to have.

00:34:26.129 — 00:35:20.620 · Speaker 1
That make sense. In fact, actually, I, I've been working on an agent insurance consortium. Right. Because in this world where we think about cyber insurance while agents need insurance to. But how do you ensure agents that fundamentally aren't humans. And so we actually just wrote a white paper about attribution attestation, being able to bind certain actions to different consents. again, all of how all of that factors into premiums.

So I don't want to take us down that rabbit hole. And coming back to, I think your points around, you know, secrets management, uh, handling and how that leads to potential risks around, uh, prompt injections. So maybe. Jeff, uh, you know, you've built, uh, secrets management, uh, platforms, uh, probably over the last decade plus.

Right? At Vault and then you build also HashiCorp Boundary. Like, how do you think about that? Right. So Fotis mentioned the network proxy method. Um, what are some that you've worked with and what are you recommending to customers?

00:35:20.660 — 00:37:48.890 · Speaker 3
Yeah, I think that the, the network proxy is it's funny because we've been building API proxies for so long. So this is you know, I think between the the very large entities out there, right, like the Z Scalers of the world down to the Boundary's that are out there. Right. Like we've been building these sorts of API gateways to secure human based access for a really long time. And machine based access. Right. Like HashiCorp built Console as a network mesh, uh, to secure and implement mTLS between services automatically. Um, I think, Fotis, one of the things that you brought up is, um, you know, agents can get prompt injected, right? And so the cool thing is, with a gateway, you can actually shut down that connection. never actually leaked any credential material to the context.

Um, the challenge that's out there, I think, is figuring out how do we recognize when an agent has been prompt injected and how do we, you know, basically this more philosophical question right now about how do we prove the integrity of an agent? Um, and that's like a level of attestation I think is sort of unsolved right now. Um, and that there's there's definitely a lot of really interesting research out there around it right now. But I agree with Fotis' analysis here. I think that having a gateway of some sort is absolutely relevant and necessary for securing agentic workflows. In the absence of a gateway, though, there's a lot of, uh, tools out there. 1Password included, obviously. So shameless plug for that. Um, because, you know, on an endpoint itself, you've got faulty technology today, right? Um, we, you know, when I do agent development on my laptop, uh, I secure my SSH keys with 1Password in our developer tools, I use our environment's capabilities. So when an agent spins up, I have to tap to give it access to an environment configuration that has sensitive material in it. I was doing I was building an

agent last night actually to to help us do some refactoring on a code base. And as I was building it in, the had access to OpenAI because that's where we were doing inference to help us understand that code base. And it was so, you know, both refreshing and reassuring to tap when it needed access to that API key. Um, and, you know,those, those methods, you know, a big part of what we're building at 1Password this year is the ability to take those methods and make them portable into other different types of runtimes, not just your laptop. Right.

Take it up to the cloud, uh, building out a SaaS based methodology that gives you that same assurance for credential management. But in all of these new environments where the runtime isn't just your laptop.

00:37:48.970 — 00:38:29.340 · Speaker 1
Um, so this is where I want to take us to the segment of our podcast where we dive into a little bit more of the details. Um, and so maybe, Fotis, starting with you. Right. We talk a lot about, you know, agent architectures. What's the right agent architecture. Um, so let's pretend we have a virtual whiteboard right in front of us. Um, walk us through, like, the agent architecture, right, schema that you would recommend to customers, especially enterprises that are thinking about building internal agents, whether it's for incident response, going through customer support, ticketing, so on and so forth. Like what? What does that look like??

Maybe starting with like who's issuing the identities?

00:38:29.620 — 00:38:31.620 · Speaker 6
Yeah, I mean it again.

00:38:31.660 — 00:40:14.530 · Speaker 4
It really depends on how, um, if that enterprise has built their own kind of, you know, uh, I guess agenetic identity Hopefully, you know, we will solve this as an industry soon, but, um, you basically need some kind of the identity provider needs to start with issuing the, you know, some kind of, uh, uh, agent ID, um, Which, um, basically you have this registration authority that issues this, um, this, you know, workload credential. But in this, um, you also will need, uh, from there on, depending on how the, the agent will be used.

If, if the, if there's let's say you have a bunch of engineers, let's, um, uh, let's talk about this, you know, incident response, kind of, um, agent. Uh, so you have this, uh, uh, this agent instantiated, and it works in your environment. Maybe you have some, uh, some some, uh, it runs within a, within a container. So you you have some done some node attestation, you can use P3 and like, you have some, some, uh, you know, mTLS where you do, uh, um, authentication using like short term, uh, certificates. So, uh, let's, let's assume you have this, you know, trusted environment. Thenyou, um,as, as long as the the when whenever the

engineer. Let's say you have a detection response engineer that wants to ask this. Okay. Like you know what I want to fetch, um, you know, logs from all these different sources because I, I need to investigate this incident, and maybe it's, um, you know, it's kind of takes me so fast to to figure out what's happening.

00:40:14.530 — 00:40:15.010 · Speaker 6
so.

00:40:15.530 — 00:43:23.090 · Speaker 4
Um, that means that now you have the, the user that is asking the agent for a specific task. So that means then the you're delegating from the user to the, to the agent, uh, this, this, uh, this specific authority. So this is the delegation we're talking about earlier. So then the agent is basically, uh, then has is authorized by the user to access these, uh, sources.

Uh, so that's also where the, you know, the, the nuances come into the picture because, um, the the agent might, uh, initially say, okay, like, this is this is the plan. So I'm planning to fetch my, you know, some some initial, you know, audit logs from from Databricks, from, uh, my, my sim from, I don't know, Amazon S3 logs and then aggregate them and then give you a report on what I, what the agent thinks it happens.

Um, but as it is as it's, you know, explores these things and kind of tries to, to understand what's happening, then it might realize, okay, uh, I actually need to fetch, uh, logs from, from, uh, from an, uh, one additional source. And then this is where the intent drift comes into the, into the picture. And so in that case, well, the question is, does the, the, the engineer need to ask, uh, needs to be asked to authorize this additional, um, I guess, uh, read on the operation from the agent, or do we have that somehow encoded in the, um, as sort of like a longstanding policy for this kind of, uh, agent?

And this is also it depends on, you know, your, use case and your risk tolerance. Um, it's, uh, there's no silver bullet. Um, so it's, uh, you can you can say that if if this agent needs to run for a very long time and it's a very asynchronous workflow, then you need to have, um, you need to minimize that sort of like human in the loop as much as possible And then in that case, what, uh, we are also exploring, and I think the industry is also exploring, is how do you have how can you instead have a model in the loop. So you have either a supervisory agent that, um, that assesses if this, if this kind of, um, additional action that the that creates this intent drift is, is legitimate enough. it basically you have like almost like a, a stochastic, uh, policy engine that kind of can reduce this human friction.

But of course, this is not foolproof. Anything that is not deterministic can go wrong. And also there is the scenario, of course, that even the supervisory agent, if, you know, if the, the, the agent itself gets prompt injected. This could directly lead to the to the supervisory agent being

prompt injected as well.

Although usually that that is a little bit harder. Um, so that's, that's uh, that's about how I, I, I envision it obviously trade offs everywhere and very, uh, dependent on the environment.

00:43:23.130 — 00:45:09.600 · Speaker 3
I'm glad you brought, uh, brought up, you know, uh, using LLMs to fight fire with fire. Right. Like, because I feel like the the world that we live in today is it's interesting. I there's a lot of this might be a little controversial, but I think that it's also pragmatic in terms of approach to start exploring these methods. The reality is um, yeah, agents are non-deterministic.

And to use them to actually apply policy to come up with the policy. Um, might sound a little scary at first because we're working with probability distributions. Um, the funny part is, a couple of years ago, when I first started exploring some of the LLM architectures out there, some of these security challenges, one of the things I recognized is that, well, if your enterprise is multiplied by every employee, multiplied by the number of agents that they're spawning, um, well guess what?

Today's enterprise, they have approvals, workflows. And at the end of that approval workflow, nine times out of ten, there's a human. And that human is responsible for saying yes or no to give you access to something. There's been mountains of research and Fotis pointed this out earlier. Uh, 95% plus of policies out there are over provisioned. Right.

We don't subscribe to Zero Trust as much as we like to talk about Zero Trust. And that to me says, well, the human approval at the end of those approvals chains is actually the least probable thing out there, right? They in themselves are a probability distribution that we should probably fear in the same way that we're fearing the agent, you know, workflows that we're trying to secure today. Um, and so to, you know, to Fotis, to your point there, I think that it's, um, you know, the research around leveraging LLM technology to develop, uh, policy based systems. I think that that's meaningful work. And I think that there's, um, you know, I think that we'll we'll get a lot out of that over time. I'm really excited about that stuff.

00:45:09.680 — 00:45:09.960 · Speaker 6
Yeah.

00:45:10.000 — 00:45:46.500 · Speaker 4
And on that note, as a small note to add, is that. Yeah. Like you mentioned, um, even in, you know, in the traditional kind of non-genetic world where we, uh, we have some kind of engineer that gets onboarded and then it's, you know, gets an initial, uh, you know, set of permissions, but then they realize they need additional access.

This is kind of the, the paradigm, but, you know, albeit very, very much slower. Uh, you still need you still have this kind of access management systems and you have the manager approve or like some, some other, uh, you know, people responsible for giving that access.

00:45:46.860 — 00:45:47.940 · Speaker 6
Um, and.

00:45:47.980 — 00:46:17.260 · Speaker 4
Again, yeah, this is this is still it's, uh, it's it's something that is very much depends on how much context the, the person needing the access gives and shares about the, you know, what is the what is the need to why do you need this extra level of access. And then the decision maker, in which case is is a human in agencies it could be an LLM. Um, how much context do they have about about what that, um, that

00:46:18.820 — 00:46:21.620 · Speaker 4
operation entails. So I

00:46:21.820 — 00:46:22.299 · Speaker 6
am I'm

00:46:22.340 — 00:47:09.270 · Speaker 4
actually optimistic that in a world where, you know, the context window is becoming much bigger, and then you also have, like this other scaffolding tricks where you can have, you know, summaries and you can basically exponentially increase the, the context window. Uh LLMs could become even, you know, better decision makers because again, they could fetch all the necessary contexts from across, like the enterprise, to make an informed decision. Andagain, you can have like, you know, different risk categories and still have a human in the loop. If, if, if we're talking about, you know, pushing to, uh, doing a, you know, pushing to prod and making a terraform change that, will that bring down everything. But um,in, in many other cases, like an LLM could be good enough.

00:47:09.510 — 00:47:29.110 · Speaker 3
Yeah. I couldn't agree more. I think that that's that's some super compelling research. I mean, the if agents, you know, if LLMs today are about 95% accurate and, you know, 95% of approvals that humans apply today are not Zero Trust. Um, it sounds like the future might be bright there.

00:47:29.270 — 00:48:21.960 · Speaker 1
Yeah. And and for me, from a, you know, auditing perspective, a lot of conversations that, you know, I have with CISOs are, well, how do you actually match what an agent's plan, right (what you intended it to do), actually match what it did. So maybe, you know, stuff that I would like to see in the future too is things around, you know, execution traces how you can match, you know, step by step, right. Potentially like hashes of different prompts, different tool calls and outputs of that agent.

Um, anyway, that's on my wish list for the next six months. Um, I feel like we could continue this conversation for many more hours. It's been wonderful having actually two identity experts in the room here, so maybe we can actually go through a quick lightning round for for both of you here before we wrap up, which is, um, if you could take a week off of work and build anything you want, what would you build and why?

Fotis, why don't you start.

00:48:23.360 — 00:48:26.360 · Speaker 4
Anything, anything I want? Okay, I, I was not.

00:48:26.360 — 00:48:35.360 · Speaker 1
Uh, I tend to be also physical. So we've all gotten answers on episodes of. I would build a chair or I would build a park bench. So just leaving it out open ended.

00:48:36.120 — 00:49:43.710 · Speaker 4
Um, yeah, I, I feel like I've been spending so much time in front of my computer that I would actually like to do something physical. Um, I don't know, I would, I would probably build a farm or like build a guitar from scratch or. Yeah, do something that is more, uh, tangible in a way. Um, I feel like with, with coding agents today, with, uh, things like Codex, we, we have accelerated software development so much.

And I think basically the bottleneck is the, the human and like the I guess what, what I think is like, how do you, how do you make sure that the code that you've written is not just, um, it does something meaningful? Because I think just because now this has been commoditized, it will become a matter of like, who has vision and like, product tastes, uh, it's going to become more, more important than, you know, just, you know, uh, writing code.

But, um, you know, back to the original question. I guess it's more of, uh, yeah, doing something in the in the physical world.

00:49:43.950 — 00:49:51.870 · Speaker 1
Well, I grew up near farms, so if you're thinking about building a farm, uh, please call me. We can do a barn raising together.

00:49:53.150 — 00:49:53.670 · Speaker 1
Keep it goingJeff.

00:49:55.310 — 00:50:20.470 · Speaker 3
Fotis, I gotta tell you, my, uh, my stepdad is a guitar builder, so. Oh, wow. Yeah. No, definitely a good selection there. Um, what would I build? Um. I love tinkering, too. In the physical world, I would probably, uh, you know, build a go-kart from scratch. That's always been kind of high on my list of something I'm going to do. like, uh, you know, mechanics and stuff, so that'd be great.

00:50:21.430 — 00:50:36.430 · Speaker 1
Wow. Well, maybe this is a trend. You know, folks in the digital world want to go actually build physical things. Um, well, how refreshing. Maybe that that's the future for us all, because agents will just completely rewrite software and do our work for us today.

00:50:36.450 — 00:50:40.410 · Speaker 4
Well, until you have robots. And then they also commoditized the physical.

00:50:41.930 — 00:50:45.490 · Speaker 1
Exactly how many years ahead do you think we are on the physical world?

00:50:46.530 — 00:51:02.330 · Speaker 4
Um, I will give you a few years to, to have robots completely take over. But, um, you know, I think it depends on the complexity of the task, obviously. But yeah, I've seen some tremendous progress on that front as well. It's it's very impressive how fast things are moving.

00:51:02.370 — 00:51:26.410 · Speaker 1
All the physical intelligence advancements. Um, yeah. That's, uh, it's just a crazy how how fast the world is moving. But, um. Good to hear you say that. At least we have a couple of years, uh, head start on that. Well, thank you so much for joining us in our virtual studio. I'm sure this space will continue evolving very quickly over the next weeks, months.

Right. I say that now instead of years. Um, so more to come.