EPISODE 1477

[INTRODUCTION]

[00:00:00] SPEAKER: The Kubernetes ecosystem has drastically changed how development teams ship software. While Kubernetes has provided many advancements in cloud infrastructure, it has also left organizations with massive security blind spots. KSOC was created to give developers and security teams a single control plane to harden multi-cluster Kubernetes environments through event-driven analysis, least privilege enforcement and remediation as code. Jimmy Mesta is the co-founder and CTO of KSOC, and joins the show to discuss modern Kubernetes security challenges and how security teams need to prepare for a future where Kubernetes is the top attack at target for adversaries.

[INTERVIEW]

[00:00:41] JM: Jimmy, welcome to the show.

[00:00:42] JMesta: Thanks for having me, Jeffrey. How's it going?

[00:00:45] JM: Not bad. So, you work on KSOC, which is based around Kubernetes security, and obviously Kubernetes has changed the world of infrastructure. But isn't container security already a solved problem?

[00:01:04] JMesta: I wish it was, yes. So, container security and Kubernetes security aren't always the same thing. So, the container itself that's running inside of Kubernetes has certain configurations, certain ways to interact with the kernel, settings, networking policies, et cetera. But there's also this whole orchestration layer, that is Kubernetes. I don't think it's solved. That's been proven time and time again, with the folks I've been talking to. There's still a future where we have a lot of stuff to figure out. At least that's the bet.

[00:01:40] JM: If I stand up a Kubernetes, cluster, and port my application to running on that Kubernetes cluster, are there automatically by virtue of being on Kubernetes new security vulnerabilities that I'm exposed to?

[00:01:56] JMesta: There could be, yes. So, starting with the notion of porting your application over to even run in a container, there are different settings that in configuration is available within that container itself, that automatically sometimes translate into Kubernetes configurations. Things like what user does the container run as? Is it privileged? What kind of networking settings exist as this container spins up inside of a pod in Kubernetes? The complexity of a Kubernetes environment, is not to be understated. Your application probably needs to talk to the API itself, or could talk to the Kubernetes API, it reaches out to other cloud services, it handles secrets. These are all things that need to be considered as you kind of lift and shift an application, and kind of plug it into the Kubernetes ecosystem.

[00:02:55] JM: Can you give me an example of a security issue that a typical Kubernetes cluster might encounter?

[00:03:03] JMesta: Sure, yeah. So, some of the ones that we kind of focus on that present a lot of problems comes down to role-based access control. Our back inside of a Kubernetes cluster, is really just how an entity, whether it's an end user, or a service account, can interact with the Kubernetes API. What objects and verbs that that particular entity can utilize to get tasks done. So, our back itself is kind of built into Kubernetes. It's its own API endpoint, its own object, and it's up to you to configure it. So, what we see often times is role-based access control being overly permissive. And what that means is a given workload inside of the cluster, maybe it has a service account that it uses a token, ultimately to talk to the Kubernetes API, and that service account has too many privileges.

So, the kind of attack vector here is somebody compromises that running workload, that container or multiple containers, and is able to grab that particular service account token presented to the API and do things almost to it, depending on the level of access at an administrative level within the cluster. There's lots of interesting examples of this where kind of web application flaws turn into something like remote code execution, turns into a cluster compromise. Our back is so flexible that it is hard to get right. And we see things like that happen all the time, and still kind of to this day.

[00:04:49] JM: So, a lot of this boils down to misconfiguration issues?

[00:04:54] JMesta: I'd say there's a lot of misconfiguration going on. Yes, there's a mix of misconfiguration and kind of CVEs, if you will. We're seeing more and more of that where there's something deeper down and kind of container runtime itself or with a particular kernel module or

something of that nature that actually gets exploited. But for the most part, configuring Kubernetes, and cloud in general is pretty tough to get right across the board. Misconfiguration is still rampant. We have CIS benchmarks. We have NSA hardening guidelines. We have different frameworks that help us kind of strive towards better configuration. But to do it right across the board is still challenging.

[00:05:43] JM: So, you work at KSOC, obviously, the company that you founded, which stands for Kubernetes Security Operation Center. Why do I need an operation center for Kubernetes security?

[00:05:56] JMesta: Yeah, that's a good question. So, the impetus for KSOC was to provide a unified control plane for security practitioners to not only gather the misconfigurations, and vulnerabilities and issues in distributed kind of multi cluster environments, but to act on them. So, as we've seen Kubernetes kind of explode over the years, it's really the backbone of infrastructure these days in a lot of large organizations. Almost be described as a data center inside of a data center. With that, you have a lot of blind spots, and security teams are still kind of ramping up to this notion of containers and Kubernetes in general, it's a very fast-moving project. The ecosystem is expanding every day, and KSOC was really born to give you that one place to go to track your progress, make sure that Kubernetes is in line with the rest of your security program. So, that's what we're working on. There are different facets of that inside of like Kubernetes, the data, we're pulling out, the types of rules we have, how we integrate with other things, but that's really what we see the need being right now.

[00:07:10] JM: So, can you shed a little bit more light on what I would get out of a security operation center for Kubernetes?

[00:07:19] JMesta: Absolutely. So, SOC is kind of an overloaded term hidden security in general. And, to us, we see a security operation center as giving you full visibility into all of your clusters, but running workloads and associated risks, and mix that with remediations through kind of a git ops process. I can talk about that in a little bit as well. The first order of business for KSOC is understanding your environment, asset inventory, where are your clusters? What regions are they running in? How are they configured kind of a cloud layer of using managed Kubernetes? Things like that, because you can't really do security on assets you don't know about.

So, we have a full automated cluster discovery mechanism that goes out, interrogates your cloud providers API's and subscribes to events related to manage Kubernetes, and gives you kind of that

high-level overview of what you have running in different environments. So, that's one piece. The second piece is any SOC, historically, would be shedding light on problems, right? Issues, whether they're CVEs, misconfigurations, best practices, custom rules, things that you're looking for, all of that is done through an event driven architecture inside of KSOC. So, we want to watch every revision of every entity inside of that cluster and run it through a rules engine to ultimately give you a collection of best practices, again, misconfigurations that you can act on. And some of those are more critical than others, and some are of the lower sort of tier that you'll deal with later. But ultimately, you should be tracking towards burning down those particular issues.

So, that's another piece of a classic SOC. Our kind of spin on this as well is to provide remediation and I think the SOC team, maybe I've kind of run SOC teams in the past, it's definitely a noisy operation, and that you're getting a flood of alerts, which ones are important, what are you going to do with them, and we have kind of piggybacked on everything being stored in git, this whole git ops notion that what you check in ultimately gets pushed out in some way, shape or form to one or many Kubernetes clusters. We are working on building git integrations that lets you start with a pull request versus just a single alert or Wiki article or something like that. So, we talk back to get repositories where the workload is configured and give you a starting point, which is the change in code that will ultimately fix the problem in one or many clusters or environments.

[00:10:16] JM: When you look at the opportunity of KSOC, and you see the growth of Kubernetes, are there particular areas where you feel you can grow into most specifically? Because if you look at like, all of the different ways Kubernetes could be deployed, you could deploy it on prem, you could deploy it in the cloud using a managed Kubernetes service, you could deploy it using a roll of your own Kubernetes strategy. There's a multitude of ways you could deploy it and there are different security issues that you could encounter across the plethora of different deployment schemas. Can you tell me what's the prototypical Kubernetes deployment mechanism? What do you offer that prototypical customer?

[00:11:10] JMesta: Yeah, it's all over the place right now. So, there's Kubernetes running anywhere you can imagine. What we've done, we're an early stage company, we have to kind of pick and choose our battles, the 80/20 rule. So, what we've seen is managed Kubernetes, being the most widely adopted for our kind of target end user, so a lot of EKs, and a lot of AKs, which is what we support today, with GKE coming very soon. That seems to be the vast majority of kind of our early customers, which is great for

us. Because those are a little – in EKs cluster, you also get a little more context into the cloud environment that you're running inside, so that's kind of why we started with this whole cluster discovery flow, because we've noticed most people are running managed Kubernetes.

As the project kind of keeps getting legs and growing, people will continue to use managed Kubernetes and that's totally fine. But we'll probably see, maybe this whole kind of resurrection of on prem and do it yourself clusters, clusters running on physical kind of IoT devices, and our installation is agnostic to any of that, given we have access to the Kubernetes API. That's for the time being, we are expanding into other product lines that may change that. But for the core pieces we just talked about, pretty much if it's Kubernetes, running a moderately recent version, we will be able to get the data we need to make decisions and remediation recommendations.

Areas that I'm also very interested in and that we're working on today are more in identity, back to our back, who has access to the cluster, what service accounts are able to talk to the API, how was that access provisioned, and that's an area where we're kind of diving headfirst into. Because quite frankly, people have asked us for it as we kind of went down these customer journeys. It's a pretty hard problem. But at the end of the day, people still access clusters and how we get to that point, how we monitor it through, kind of watching logs, how we detect anomalies, that's all to come in kind of a KSOC ecosystem as a whole different product line that we're really excited about.

[00:13:38] JM: If I have an incident on my Kubernetes cluster, how does that get detected? If somebody has infiltrated my Kubernetes cluster using, for example, some certificate that they stole, or credentials that they stole, how do you detect and exile that attacker or respond to that attacker?

[00:14:03] JMesta: Yeah, in that particular scenario, let's say you checked in a certificate or some sort of authentication token for simplicity sake, and to get in somebody took it, and your Kubernetes, you couldn't say it was checked in and your Kubernetes API was sitting on the internet, you could present that credential to the API itself, authenticate and do what you need to do from an attacker's perspective, at least start poking around giving you our back configuration.

So, detection in that scenario is really hard, because the only real place that tells the story would be your Kubernetes API audit logs. And I would say to this day, they're underutilized from a security standpoint. Again, we're hoping to change that. That's an ingest stream for us as well, in our product.

So, you see a very rich kind of log history of who's doing what against the Kubernetes API, and you would have to very intelligently be able to tell, did this credential come from a trusted place? Luckily, we have a lot of kind of SSO, IDP integrations now that can force MFA or short-lived credentials. But if it's a hard-coded credential, you're going to have to look at things like within that log entry, its source IP address, which is obviously very loose and being accurate or detectable against anything mixed with deny statements.

So, is that person getting denied by the API for the type of action that they're performing? And then you'd have to generate an anomaly that says, “Hey, this looks like a suspicious user, we need to automatically revoke this particular credential.” I think we have a long way to go until we get there, unfortunately. And keep your Kubernetes API off the internet, I guess too, is the other helpful part of that. It's not foolproof, but it's definitely helpful.

[00:16:13] JM: Is that a regular problem? People leaving their Kubernetes API access open on the internet?

[00:16:18] JMesta: Oh, yeah. I think it is kind of the default and kind of the managed Kubernetes environments, if you want to get up and running quickly. If you're at a large established organization with a proper kind of Kubernetes story, probably not. But it's still all over the place in terms of like, what's open in the internet and not. It's, I think, Rory McKeon, I think it was him. He went on census.io, just like last week, or something and was like, “Hey, here's like, a million Kubernetes API endpoints open to the internet still.” Does that mean they're vulnerable? Maybe not, but you're definitely setting your attack surface up for failure if you keep those things sitting on the wider internet.

[00:17:06] JM: There's a term you've probably heard called crypto jacking where somebody malicious manages to install Bitcoin mining software on your cluster, and then it just runs there and mines Bitcoin. Is that a frequent problem for Kubernetes clusters?

[00:17:28] JMesta: Yeah. I don't know why crypto mining, crypto jacking has become kind of the de facto attack inside of containerized environments. I mean, I guess I have a hunch, it's because it goes largely undetected, mixed with Kubernetes actually being a pretty good place to mine cryptocurrency due to it’s kind of like these nodes are fairly beefy, auto scaling, and I think it's still a problem. It's more of a drive by sort of thing, opportunistic sort of attack vector. JW Player, their engineering team, they put

out a really good kind of post mortem article. This is probably two years ago, and it was titled something like, “How A Cryptocurrency Miner Made Its Way Onto Our Internal Kubernetes Cluster.” It really outlined the problem that we have with misconfigurations.

So, kind of the attack path really was somebody spun up a network observability utility inside of the Kubernetes cluster. The SRE team said, “Hey, I'm going to purchase or use this tool that's going to help me gain some insight into my cluster.” They installed it into the cluster and what ended up happening is, they probably didn't do a full code review of all of the manifest that generated this particular utility, and it had a service with a type of load balancer, which really is telling the Kubernetes API, especially if you're in EKs, or managed environment, go give me an external load balancer and a public IP address.

What happened is this utility spun up a load balancer, and ELB, or ALD inside of AWS with a dashboard attached to it, for the ops team to use internally. And somebody found it on the internet, and this dashboard had a built-in shell, like its own little utility in the browser that you could use to kind of troubleshoot some things and gain access to the cluster. There was no authentication on this dashboard. It was never meant to be on the internet and somebody came in and found, it got to the shell, realizd that this was a privilege pod, meaning that there was really no kind of classic like restrictions on what system calls can be made from this process inside of the cluster, and they were able to ultimately escape out of the context of that container, which was this dashboard, and they were root on the node itself, that was running that pod.

Once they were on the node, they installed cryptojacker.sh whatever the malware was, and they planted it there and left it, and that means Kubernetes wasn't really aware of it, and the only way they found out was through like a Datadog alert. This node 37, or whatever it was, was pegging CPU at 100%. They looked into it and realized it was this pod was highly misconfigured. It had tons of access to do whatever it needed to, and somebody took advantage of that. So, yes, that's a long-winded way of saying crypto jacking is still a thing, it will continue to be a thing, until we kind of locked down our configurations.

[00:21:03] JM: Do you think like, from the – I mean, this is getting a little off topic, but from an ROI perspective, is there something better that an attacker could do with all those spare compute cycles across a Kubernetes cluster?

[00:21:19] JMesta: It depends what their intention is, right? I think Kubernetes is a good point to bounce off into other parts of your cloud environment. If you are looking to steal data, and you get into a Kubernetes environment, access to the API, you can dump environment variables, ultimately, your pivot into like an RDS database or your pivot into KMS, it's there. You're one step away from being able to actually cause some pretty severe damage from like a data leakage perspective.

I do believe cryptocurrency mining is kind of the lowest barrier to entry. So, the ROI might be seemingly low, right? If you install it on one node, like what are you actually getting out of that? You're just really hoping that nobody catches you for some extended period of time. But the real ROI for me and like, when you get into more, I guess sophisticated attackers would be using Kubernetes to pivot elsewhere, dump data from this table in this database to go grab an S3 bucket. Because once you're inside, on a node, you're going to be able to see and collect a lot more.

So, if you're very targeted, and you want social security numbers, and you know they're potentially in this database somewhere, you would probably use Kubernetes as their launch point, and you would have a much higher value, if that was your target. But crypto is easy and you can almost script it, like this stuff's quasi wormable, right? You could probably write a worm or some sort of scanner that goes and finds this pattern of exposed dashboard, you know that this thing is open, you dump this payload, you elevate privileges that can be automated 99% of it. So, I think that's why it's so common.

[00:23:10] JM: One of the ways that you secure Kubernetes clusters is with entitlement management. And if you look at entitlements, can you describe what that actually means for a Kubernetes owner? So, if I'm trying to have my infrastructure, managed in a way that has the privileges, the entitlements over my infrastructure properly managed, what does that look like? And how is that configured for a responsibly managed Kubernetes cluster?

[00:23:49] JMesta: Yeah, this kind of starts it in the cloud in IAM world, right? It's an ongoing discussion point problem, what have you, even inside of cloud environments where you have IAM policies that may not match the desired state of least privilege, whatever that may be. Least privilege is kind is a hope and a dream in a lot of environments. So, inside of Kubernetes, what we've learned is humans are still talking to the Kubernetes API, and our back is messy enough to throw your hands up in the air and give that person or that entity too many entitlements to do too many things inside of the cluster.

So, we look for things like who is bound to the cluster admin built in cluster role inside of Kubernetes? Because cluster admin is like stars across the board. That's a bad start, typically. And our vision for this in the future really boils down to better logical grouping mixed with just in time developer access for humans. This is a little different in service account land where there is not a human. But if I can have some other source of truth, validating my authentication, I have MFA. I've used Okta. I've used, whatever it is, and I've proven I am who I said I am, how does that kind of transpire into Kubernetes through authorization? What group am I in now? How do I get access to the things that I need, and nothing more?

So, what we're working on is that kind of just in time provisioning of that access, mixed with watching what that entity is doing, right? If I can kind of parse through the logs of Jimmy over some amount of time and see that do Jimmy has cluster admin or close to it, but he's only ever accessed these resources in this namespace, we can come up with a closer to least privilege kind of policy for that individual or that group of individuals. So, I think like, people tend to kind of drop off at IAM. We're getting better at like AWS IAM and GCP, and these things, and Azure, and then kind of just dump people into Kubernetes, and let them do what they will or leave it pretty wide open. So, that's our aspiration is what we're building today is a way to provision access, but gets you closer to that least privilege nirvana.

[00:26:35] JM: Can you describe in more detail how you actually hook into a Kubernetes cluster?

[00:26:43] JMesta: Yeah. So again, the cluster discovery talked about that, that's just your cloud API. And there's many ongoing conversations from vendor land us being included of like, what is the best way to actually get the data we need inside of the cluster, and there's opinions all over the place. We have taken the approach that is – there should be least amount of friction as possible with the operators of the cluster. Performance is a big deal for us. So, we subscribe to Kubernetes API events. We see if a deployment gets changed, the revision is bumped and you pushed out a new version of your microservice. We see that and then we take that data associated to the greater entity itself for that deployment in this example, and we look for misconfigurations or issues with that particular revision. With that, we're a deployment ourselves. So, we're not a daemon set, or like a sidecar. We use a single deployment, and we get the data we need there. In the future, there will be other plugins that may

trickle over into things that are closer to daemon sets, as you get into eBPF and those things, but that's not what we're focused on today.

[00:28:04] JM: Yeah, so mentioning EBPF, it does seem like a lot of the security vulnerabilities that can emerge involve a network connection. Can you distinguish between network vulnerabilities and vulnerabilities that are more about the standing container themselves? Are there some network vulnerabilities that you could enumerate?

[00:28:32] JMesta: Yeah, I mean, eBPF is an amazing Linux kernel feature. We're getting very deep insight into things that are happening at a pretty low level and that's useful. There's no doubt about it. I think there's – eBPF is not a Kubernetes feature. So, there is this mapping that needs to happen of like, what's going on with this container that's part of this pod, and it's in this replica set, and in this cluster? There's mapping that needs to happen there in eBPF. The types of data that you can pull out of eBPF is certainly more at the runtime perspective. There's an interactive terminal session opened or this networking poll happened from this workload to this workload. Those are things that are really useful for kind of a reactive, sort of like somebody's in my cluster doing things I didn't expect them to be doing, and there is definitely a place for that. The difference is when you look at kind of the Kubernetes manifest, mixed with the logs that are available, mixed with our back policies, you tell a different story and you try to fix any potential problems at the source through better configuration. So, there's a world where both of those live together and it is absolutely aware hope that we will involve in. We're just starting our journey with the configuration side, because of the lower impact.

eBPF typically runs as a daemon set, meaning it's on every single one of your nodes. It's performant, but that doesn't mean it's free of costs from like an operational standpoint. We want to make sure that we're providing value to security teams very quickly, and you're prepackaging some, I would say, quasi opinionated packs of rules for them to get started. So, we're taking the configuration route and that's turned out to be a good choice for where we're at.

[00:30:42] JM: When you think about the design of a Kubernetes security control plane, you could have designed it such that agents are sitting on all of the Kubernetes clusters, and doing various processes while they're sitting there. Why didn't you take the agent-based approach?

[00:31:06] JMesta: Yeah, a few reasons. Even before KSOC, I was working with big companies kind of helping their Kubernetes security story, and kind of some of the feedback I was getting was like, we don't have the appetite to introduce performance, whether it's CPU or memory, or just regular or just distribution of these agents. That is a nonstarter at some organizations. I would say a lot of them. So, we decided to make our presence in the cluster as simple as humanly possible, and put the burden in our side, in our kind of SaaS, ETL platform, and that seems to be going over really well. Because ultimately, we have the ability now to iterate faster, have less kind of these – I used to work for a company called signal sciences and we had agents, right? We were a web application firewall. So, we had agents scattered throughout and pushing new rules or new features out to these agents, that was my core responsibility. My team became very difficult, right? It's not easy to have major agent kind of revisions.

So, the more simple we can make our presence in the cluster, the better. And we chose that out of being able to iterate quickly. We want, like a CVE to come out or some issue that your kind of research team detects to build the rule, the policy, et cetera, push it out, all of our customers get that out of the box on day zero, essentially without changing everything that's going on under the hood. So, it is one way. I think it adds other complexities on our side, from an engineering perspective, there's no doubt. We have a lot to handle the networking and throughput and data. But it's turned out to speak to security teams who don't always have the leeway to bring heavy agents into these environments. So, it's been a success there.

[00:33:26] JM: There's like a ton of tooling and projects that have come out around the Kubernetes ecosystem, and I think the sales process is probably pretty hard because any large enterprise gets pitched by a million different Kubernetes tools every week. What has been the selling point? I mean, for example, like Aqua Security has been around forever, well, since at least since the beginning of the Kubernetes security world. They have considerable footprints. There are lots of security vendors in the ecosystem already. First of all, how do you manage to get your foot in the door? And how do you compete for security related contracts?

[00:34:18] JMesta: Yeah, you're not wrong. There's more noise than definitely would like to have in this particular arena. Our approach really, it's really not to like go after any of these other technologies in particular. There's actually something to be said, like, we could run alongside some of these, right? Image scanning, for example. A lot of people buy CVE scanning tools and plug them in and they're

good to go. But that's not exactly the area we're targeting. We will support that feature and some integrations there but I think as far as like, I know not got to call on any direct competitors. But we've really kind of are focused on the security team first, like, that's our network, that's where I kind of have been working forever. I mean, my co-founder, and we focus on ease of use basically. We have had a UX engineer from the first day of KSOC's inception, brought on somebody who's done a lot of consumer stuff, and B2B stuff and datacenter things.

We are very focused on getting up and running quickly without having to know any specific real language or under the scenes configuration. These things should be self-service, as much as possible. That mixed with being able to fix the problem at its source and git, what we're working towards really hard, is huge. I think we've hit this inflection point with security tooling, where it's like, "Stop sending me alerts. I'm over it. I can only handle so many Jira tickets. Show me the code that needs to be fixed." So, that's again, because of the way we're architected, we have a lot of visibility into that and what the fix is, and should be, and our teams built like a lot of our engineering team has a very strong kind of platform engineering background, and we want to appeal to platform engineers who have to run this stuff day to day and deal with the pull requests and deal with the installation.

I think just an overall user centric flow that addresses problems that like a multi cluster layer, that we treat every company like they're going to have not one cluster, where they're installing one thing, it's like always multi cluster across multiple clouds, and that's kind of our UI, is centered around your entire suite of Kubernetes clusters, and I think that's pretty unique. So, we haven't had a lot of friction with – we buy this other thing, so we aren't going to buy you. It'll come, but you know, we're early and we're building for the future of how people are going to use this thing.

[00:37:11] JM: What have been the hardest engineering problems when building KSOC?

[00:37:15] JMesta: Because we are not agent centric, we've had to really level up our data processing kind of architecture. We're kind of heavy Kubernetes users ourselves. That being said, we have microservices scattered about. Getting data from point A to point B, choosing the best, most performant and like, at times cost effective way to do that. That's been challenging. Honestly, at a kind of human perspective, it's hiring. I think there's a unique skill set we're looking for, and we're getting in the groove of that now. But finding awesome talent and people who want to dive in has been challenging. We also are distributed across many, many time zones at this point. As a young company, from an engineering

standpoint, it is hard. We're a remote first distributed team, so we've had to really grow up fast when it comes to being very diligent about our engineering procedures, our sprint planning, everything that comes – the nitty gritty of how things get done and prioritized across the span of time zones, is not easy for a startup. So yeah, I would say those are the big ones.

[00:38:38] JM: Do you find that the heterogeneity of different Kubernetes deployments of accommodating all those different deployments is difficult? Or is there a consistency to the API and integration points that makes it simple enough to integrate with the plethora of different types of deployments?

[00:38:59] JMesta: We're taking a pretty logical approach to what we support and when. If we're talking about like, custom resource definitions and kind of one off very custom Kubernetes implementations, we support whatever the API has available. But we have not been hit with anything that we couldn't handle yet. I'll probably eat those words later. But yeah, I think the Kubernetes API luckily, offers kind of a standard interface for us to interact with. As we expand the product lines, we'll have to deal with wildly different problems. But again, it's like you're working in the web space, which I was at before, that was pretty tough, because you're just dealing with HTTP requests. We don't know anything about the back end. These are just like – you are basically just proxying the internet and everything about it. I would say Kubernetes is much more structurally sound than that sort of defense. Mechanism, so we have a clear path through these predefined API's that are well documented to get what we need and do what we need to provide value. So, I think time will tell again. But for now, feeling pretty good about the standardization of Kubernetes, at least in terms of security.

[00:40:22] JM: Cool. As you begin to wind down, maybe we could talk about your vision for the future. What's the next most acute pain point that you're planning to work on in building your Kubernetes security suite?

[00:40:39] JMesta: Just in time access is a huge undertaking. I also think striving towards least privilege providing mechanisms to get there for people. We've actually had, like a lot of great conversations around that, and we're building something pretty cool there. Those are immediate term. The future is what we're hearing and what I tend to agree is like Kubernetes, philosophically, I guess, it sits in your cloud, typically, but it's not self-contained. So, a cluster or a pod within a cluster is out talking to RDS or KMS, or uses IAM to do this thing. So, there's this like Kubernetes at the nucleus of

the problem. But how do we apply security and observability to the other parts of the cloud that this nucleus that is Kubernetes is touching? And I think that's the future for us is really like, let's get the blast radius kind of under control, and understand how far this thing reaches out.

We intentionally are setting ourselves up to do that with, again, like we already tied into the AWS API and all these other things. That's a natural progression for us. Some people are even asking, like, “Hey, KSOC, can you just go the central point of cluster, create cluster management?” I just want a secure cluster. Let me click a button. Let me run a Terraform module. Let me do something to spin up a cluster that I know is going to be within the constraints of my security policy. It’s an interesting problem as well. I think that's something we’re at least experimenting with today.

[00:42:23] JM: Yeah, I mean, that last part that sounds like kind of out of scope, right? Like spinning up clusters?

[00:42:28] JMesta: Maybe not. It's probably more of like an integration sort of play, where you're piggybacking on some other cluster management ecosystem, and you're helping overlay security on top of it. That's probably the better play. I don't think we're going to be in the kind of a cross plane sort of even Terraform style implementation. But people are curious. And then there's always this mix of AppSec, and cloud security. It's like, I want to know about my application security, inside of these environments, and who's talking to it through network? What networking, protocols and ports are open? That's a different fuzzier area, but it's also interesting to kind of blend the two sort of concerns together.

[00:43:16] JM: Cool. Well, real pleasure talking to you and it seems like you're off to the really good start with KSOC so far. I look forward to seeing where the company takes you.

[00:43:25] JMesta: Thank you. Yeah, it's been fun so far. I really appreciate being on the show. I’m a longtime listener, so this is exciting for me. So, thanks again.

[00:43:34] JM: Thanks Jimmy.

[END]