EPISODE 824

[INTRODUCTION]

[0:00:00.3] JM: A Kubernetes instance occupies a wide footprint of multiple servers creating an appealing target to an attacker, due to its access to a large pool of compute resources. A common attack against an exposed Kubernetes cluster is to take it over for the purposes of mining cryptocurrency. Thus it is important to keep a cluster secure. The importance of security is magnified for a cloud provider.

A cloud provider runs a managed Kubernetes service, which might be running thousands of Kubernetes clusters. If the cloud provider's chosen distribution of Kubernetes contains a vulnerability, or if the Kubernetes instances are misconfigured, all of these clusters could be exposed to the same common vulnerability.

Maya Kaczorowski works on the security of Google's managed Kubernetes service GKE. In today's show, we discuss the attack surface of a managed Kubernetes service. Maya was previously on the show to talk about container security. This episode is a good companion to that one, as well as a previous show we did with Liz Rice about container security. We now have covered container security, Kubernetes security and container platform security.

The FindCollabs Podcast is out. FindCollabs is a company and a community I started recently. The FindCollabs Podcast covers some of the goings on in that community. FindCollabs is also hiring a React developer. We are looking for sponsorships for Q3 for Software Engineering Daily. If you're a company that wants to reach 30,000 developers, maybe many more, we actually don't have great statistics on how many listeners we have for complicated podcast infrastructure reasons. If you are interested in reaching a large number of developers, you can go to softwareengineeringdaily.com/sponsor.

Let's get on with today's show.

[SPONSOR MESSAGE]

[0:02:10.2] JM: Testing a mobile app is not easy. I know this from experience working on the SE Daily mobile application. We have an iOS client and an Android client and we get bug reports all the time from users that are on operating systems that we did not test. People have old iPhones, there are a thousand different versions of Android. With such a fragmented ecosystem, it's easy for a bug to occur in a system that you didn't test.

Bitbar is a platform for mobile app testing. If you've struggled to get the continuous delivery in your mobile application, check out bitbar.com/sedaily and get a free month of mobile app testing. Bitbar tests your app on real devices, no emulators, no virtual environments. Bitbar has real Android and iOS devices and the Bitbar testing tools integrate with Jenkins, Travis CI and other continuous integration tools.

Check out bitbar.com/sedaily and get a free month of unlimited mobile app testing. Bitbar also has an automated test bot, which is great for exploratory testing without writing a single line of code. You have a mobile app that your customers depend on and you need to test the target devices before your application updates rollout.

Go to bitbar.com/sedaily and find out more about Bitbar. You get a free month of mobile application testing on real devices and that's pretty useful. You can get that deal by going to bitbar.com/sedaily, get real testing on real devices, get help from the automated test bots so that you have some exploratory testing without writing any code. Thanks to Bitbar. You can check out bitbar.com/sedaily to get that free month and to support Software Engineering Daily.

[INTERVIEW]

[0:04:19.7] JM: Maya Kaczorowski, you are a Product Manager in container security at Google, welcome back to Software Engineering Daily.

[0:04:24.8] MK: Thank you so much for having me again.

[0:04:26.4] JM: The last time we talked, I believed that the things that were at top of mind in terms of container security issues that you were concerned about, that you were addressing were things like, I've got an insecure Kubernetes cluster and somebody accesses it and starts

mining Bitcoin, that was one thing. The other thing I remember is people can – if your containers are insecure, perhaps people can jailbreak out of the containers. It's been, I think about a year since we last talked? How has the world of container security evolved since then?

[0:04:58.9] MK: Yeah, that's a great question. In the last year, we start to see the first couple of real attacks on Kubernetes in the wild. First in late 2017 and early 2018, Tesla, Aviva and Weight Watchers all had exposed Kubernetes dashboards that had credentials on them, so these were not password protected. Attacks are able to go in and grab those and start mining cryptocurrency exactly as expected more of the same types of attacks.

We saw also Shopify had a researcher report an issue to their bug bounty program, where their cluster had access to metadata that it shouldn't have had access to, knowing metadata. The researcher was able to have control over their whole cluster, which was not expected. We also saw malware embedded in images on Docker hub. The handful images on Docker hub that had cryptocurrency embedded malware in those images. Still seeing some of the things that you just described around having cryptocurrency mining is still very much what attackers are going for, any way to make money, which is bots or cryptocurrency. I'm not really seeing in the wild, live attacks of container escapes in yet.

[0:06:00.6] JM: We actually did a show recently with a company called Safe Tulpa that is working on – basically their entire business focus is this cryptojacking issue. How big of an issue do you think this is going to be for the industry?

[0:06:15.0] MK: I think it's the primary way that attackers have to make money if they can gain access to cloud accounts today, right? If you think about what people historically would have done, right? They –

[0:06:24.2] JM: That by the way is a huge statement. I mean, I believe you. It's just that is a big statement. I didn't realize. I mean, that's news to me.

[0:06:31.7] MK: I think if you think about what attackers are historically going for, right? They just want to monetize what they have access to. 10 years ago that might have been stealing a credit card and selling it in the black market and making some profit from that, right? Five years

ago was building a botnet. Today, mining cryptocurrency is probably the easiest way to make a lot of money online without people really noticing and being able to trace it back easily to you.

[0:06:51.3] JM: It's like just open source script, right? It's you just install this little script thingy, maybe you obfuscate it a little bit and then you're instantly mining cryptocurrency on somebody else's infrastructure.

[0:07:02.5] MK: As soon as you can gain access to it, you can install something, yeah, for sure. I think that's much more – that's something else people don't realize is they think people are going after anything, but also Kubernetes to try to get to your workload. Nobody cares about your workload. Nobody's actually trying to get to your user data. They just want to access your compute. It's not about the user data that's actually on that workload yet. If somebody is going for that, then they're actually a rather sophisticated attacker and it's a targeted attack that's going after that particular workload.

[0:07:27.7] JM: Tell me more about the dynamics of cryptojacking as you see it.

[0:07:30.9] MK: I don't know. I think it's going to keep happening until we find a way of stopping it, right? It's typically somebody will download some shell script that will execute, like you just described a small binary, usually something that's open source or easily available. I might have bought something a little more sophisticated on the black market. Then I just funneled that back to my wallet somewhere. It becomes very, very hard for me to trace that. Even if I can, the damage is already done. I already used up a bunch of compute.

[0:07:56.2] JM: If I'm operating a cloud, or if I'm operating a Kubernetes cluster as an enterprise, is there any way to scan my clusters to know if this script is running on it?

[0:08:10.2] MK: Yeah. I think there's a couple things you would might want to look for. You would look for – or a couple of different ways you could start to detect this. You could look for on public clouds if your billing suddenly goes up all of a sudden. That's a typical way that this is found on your –

[0:08:22.6] JM: Pretty good to bug?

[0:08:23.9] MK: Yeah. If you have something using monitoring of your CPU usage, that's another good way of looking at if it suddenly spikes up, something could have gone wrong. Then there are known IP addresses for Bitcoin miners, etc., at wallets. If you're connecting to any of those known addresses, then you know that something might –

[0:08:39.3] JM: Just scanning the network command connectivity.

[0:08:41.9] MK: All of these again are assuming relatively unsophisticated attackers. I mean, if I look at those Docker hub images, what they did really well was they were barely using any CPU, they're embedded in images that looks like they were harmless and were being used anyways.

[0:08:56.7] JM: How did they actually get caught? Was somebody just looking through the container images?

[0:09:00.5] MK: Yeah, a researcher, I believe it was ChromeTech looked through and found, I think it was 17 images. They look like play images. They were called things like Docker 123321. Obviously, things people were just rolling out to test things, hopefully not rolling out into production. I will not quote the number, because I don't remember it offhand, but it was on the order of millions of times that it was downloaded, these images.

[0:09:22.5] JM: When that happens, is there any way to notify the people who hey, by the way, you are mining crypto –

[0:09:28.5] MK: Yeah, that’s a real problem.

[0:09:30.2] JM: Is it?

[0:09:30.8] MK: Yeah. What you're describing is this problem that we're seeing starting to emerge more and more, the software supply chain, where if I download an image from Docker hub, I don't really know where it comes from. I don't really know whose binaries are in it and I don't know who's running it at the end of the day. I don't actually know who pulled the image,

what area it ended up in, etc. This concept of a software supply chain similar to a hardware supply chain, but for software, is to know where your binaries come from, where your dependencies come from, where your configs come from and who has the authority to change those before they end up in your environment? Then once things are running in your environment, how do you know if you're actually affected by a new vulnerability? How do you know what's actually running?

[0:10:05.7] JM: Is this is a new problem, or has this been – there's a pseudo app to get command, I always used to run when I actually programmed computers. That's a thing where you get packages from some distribution system. Has this always been a problem?

[0:10:22.3] MK: I think it's always been a problem. It's something that attackers are starting to exploit more, or are seeing as an opportunity to exploit more. I mentioned Docker hub. There is also the NPM security issue in about September, where similarly package manager had a user maliciously install – a user had access to maintain the repository and maliciously installed again some cryptocurrency mining wallet-type software.

At the end of the day, this is just due to the proliferation of software that exists in our infrastructure today, right? The fact that so many different people are using so many different open source components, right? If I look back at something like heartbleed, which was five years ago now, open SSL is so widely used. Imagine if somebody purposefully put something bad in open SSL today, it would end up everywhere, right? There's no controls over where that goes, because there's actually security in numbers and it's good to have more eyes create shallower bugs, etc. At the same time, you need to know where that's actually coming from and who's responsible for it.

[SPONSOR MESSAGE]

[0:11:27.9] JM: To get ahead today, it helps if your organization is cloud native. The 2019 Velocity Program in San Jose, June 10th through 13th, we'll cover everything from Kubernetes and site reliability engineering, to observability and performance, to give you a comprehensive understanding of applications and services, and stay on top of the rapidly changing cloud landscape.

I attended Velocity conferences early on in Software Engineering Daily and it helped me grasped the times and the fast-pace of change in today's times. There are twin revolutions in open source software and cloud. To understand the nuances of these twin revolutions, going to conferences can really help.

At the Velocity Program in San Jose, you can learn new skills, you can learn technologies for building and managing large-scale cloud native systems, you can connect with peers to share insights and ideas. Join a community focused on sharing new strategies to manage secure and scale the fast and reliable systems that your organization depends on. You can get expert insights and essential training on critical topics, like chaos engineering, cloud native systems, emerging technologies, serverless and much more.

Listeners to Software Engineering Daily can get 20% off most passes to Velocity by using code SE20. You can go to velocityconf.com/sedaily and use discount code SE20 on bronze, silver and gold passes. O'Reilly Velocity Conferences are a great way to learn about new technologies. I can speak from personal experience on going to those conferences. You can go too. It's San Jose, June 10th through 13th. Go to velocityconf.com/sedaily and use discount code SE20 to support the show and get 20% off. Thank you to O'Reilly.

[INTERVIEW CONTINUED]

[0:13:47.9] JM: Are you surprised we haven't seen more, like Mirai botnet level catastrophes happen across the industry, given the – Whenever I have a conversation with a security person I'm like, "Oh, my God. I'm so worried about the world." In actuality, you don't seem to see too many things that I wake up and I'm like, "I'm terrified today. Mirai botnet is attacking a DNS provider."

[0:14:11.0] MK: Maybe you should be spending more time with security people. I don't know.

[0:14:13.7] JM: Really? Should I be more afraid?

[0:14:15.5] MK: I don’t know that it’s substantially worse than it was five years ago or anything like that. I think the threats are different. Attackers are evolving, but so are the defenders. We're not net losing a battle or anything like that. The attacks are certainly getting more sophisticated and more targeted. For an average user, an average developer, an average enterprise, unless you're being specifically targeted, it's still mostly drive-by attacks, right?

I'm looking for an application that you're running that's sitting with a public IP address, that has a known vulnerability, like Apache struts left unpatched for many years. Then I'm just getting in and doing something harmless. Relatively harmless.

[0:14:50.6] JM: Right. That was the other recent one that freaked me out and made me look up what is my credit score actually do? Does this information actually relevant to me? Can I even do anything? Turns out the answer was probably not really. I don't know. I froze my credit score. I don't think anything happened.

[0:15:09.7] MK: You're better than me. I don’t think I did anything.

[0:15:12.1] JM: Yeah. Actually, no. You probably didn't waste your time is what happened. To come back to the cryptojacking issue, what I find interesting about the cryptojacking space is even if you – you could compromise somebody's infrastructure, but not actually mine the currency directly on the infrastructure. You could send a blob of JavaScript to all of the – the front-end applications that are consuming that application, and you can mine cryptocurrency in their browser. Have you seen that attack?

[0:15:43.5] MK: Oh, on user browsers. Yeah.

[0:15:45.2] JM: I mean, I guess that's not really – See, that's interesting because that's a vulnerability for the cloud provider, but it only affects the end user. Is there any way to scan for that?

[0:15:56.0] MK: I don't know offhand. I would look for the same types of web app vulnerabilities that I would have elsewhere. You're looking at basically script injection to the browser.

[0:16:04.3] JM: Right. Yeah, interesting.

[0:16:06.2] MK: I want to go back to what you mentioned earlier on container escapes. We haven't seen that in a while. We have had vulnerabilities more recently that have exposed, that is a potential attack method, right? That RunC had a vulnerability about six weeks ago now. That was exactly this. It was a container escape.

I think we're seeing a lot more interest and excitement in this, because people don't understand that containers aren't actually very good at containing. They're really good at making applications portable, but they're not meant to be this, what you called I think, a jail effectively. That's not their goal. There are new projects like gVisor, Kata Containers, [inaudible 0:16:42.2] etc. filling that space.

[0:16:44.1] JM: What do they do that lets them have better security properties? Is there anything specifically that they're containing?

[0:16:52.6] MK: Effectively, you're just restricting the set of calls that the application can make, to make fewer risky application calls. If a file read, or file write was particularly susceptible, then you might want to say, “No, I don't want you to do this in my environment.” You're less likely to have an attacker be able to exploit that to use it to do something bad.

[0:17:10.6] JM: In working on container security at Google, are you specifically in the Google cloud division, or are you also working on Borg containers security?

[0:17:19.5] MK: Yeah. I primarily focus on Google Cloud.

[0:17:21.3] JM: Google Cloud, okay. Do you have conversations with people in the Borg area, or does that even matter? I mean –

[0:17:29.0] MK: Definitely. I mean, Google Security is effectively one security team for both Google and Google Cloud, so it covers all aspects of security.

[0:17:36.9] JM: Tell me about some of those conversations. What is a lunchtime conversation between two container security experts?

[0:17:43.0] MK: I think some of the things that we think about a lot and worry about a lot is having multiple layers of isolation. We just talked very briefly about container escapes and whatnot. The idea being you should never have to only rely on one layer to get security, right? The attacker should have to have two different sets of things that fail in order for them to actually successfully compromise something.

That applies to container security, but that apply – and for container isolation, but that applies in terms of any types of security at Google. If I think about encryption for example, we encrypted data at the device layer and then again, at the filesystem layer. We encrypt data in-transit at the network layer and again, at the application layer. I need two separate controls to fail in order for something to then be broken and for the eye to talk and to have access to something. That's the thing that how do you build that into containers, into the newer technology that we're developing now?

[0:18:31.4] JM: Do you focus specifically on container security right now, or is there also the Kubernetes control plane security that you're – that you think about?

[0:18:40.0] MK: I think about both, there was a handful of product managers focusing on all of these areas. It's not exclusively me.

[0:18:45.6] JM: Yeah. Do you think of security as something that in terms of container security, are you thinking of it more in terms of this is something where I want to offer a granular set of features to the end-user, or are you thinking of it more as this is – security of your containers is something that should just silently exist?

[0:19:07.0] MK: I'm leaning more towards the latter. It’s not about it being silently existing, it's more about the users shouldn't have to make a lot of those decisions, right? We should give the user same defaults and strong configurations, so that they don't have to go configure something if they never quite getting around to doing it.

If I look at GKE for example, some of the configurations that we made in GKE have changed and lost a year or two, to have those stronger defaults, things like having our back on by default, things like having the Kubernetes dashboard disabled by default. New changes to not having basic Auth and client certs created for new clusters. These are small things that can be then exploited by an attacker that are unnecessary, right? You're creating a larger surface of attack without having a good reason for doing so. That's ultimately not a good security principle, right? You want to balance security and usability.

[0:19:55.6] JM: In some areas of security, you have an uneven publicity that you want to expose for a given issue. If an issue is if you discover a super sensitive issue as a security researcher, you don't want to just go publish it on a forum somewhere, because instantly, a bunch of people are going to get taken advantage of. Obviously, that comes at odds with the whole open source aspect of it. How does that balance get struck in the Kubernetes and container community?

[0:20:25.9] MK: Yeah. We're describing is what's referred to as responsible disclosures.

[0:20:28.7] JM: Responsible disclosure.

[0:20:29.6] MK: If I'm an attacker, or I should say not an attacker. If I'm a gray hat researcher, I shouldn't necessarily go tweet about the new thing that I found in the new O day. It's my responsibility to go first tell the people who are affected that they're affecting. Give them a chance to develop a patch, respond, etc. If I've gotten zero response from them, if they've told me some ridiculous timeline in which they're going to patch it, etc., something that's not acceptable, then I might go and make it public to get more attention and try to get that fixed.

If I look at Kubernetes, Kubernetes actually is a very good vulnerability response system, especially for an open-source project of its age. It's doing incredibly well. There's a product security team that will take any incoming vulnerabilities that get reported by the community. The product security team evaluates those, creates a patch, this is all under embargo, so this is not done in public. Communicates back to the researcher the timeline, etc., that they're working on. That's actually one of the key pieces, I think a lot of programs miss.

You want to be able to tell the researcher that what they did matters, right? Yes, we listen to and we're working on it. Not like, give them complete silence for a couple of months. Develops a patch on under embargo, distributes the patch using the private distributors list. Anyone who's a official distribution of Kubernetes will get this patch ahead of time. Then the embargo lifts and patches tend to roll out at that period in time. Users are notified and the patches made available in coordination.

That's what's happened with probably about a dozen vulnerabilities now in Kubernetes; companies will send a bulletin to the Kubernetes security announce mailing list with all the details of the vulnerability, where the patch is available, how to get the patch, etc. Most major providers now are updating their own security bulletins to have that information as well.

[0:22:11.7] JM: As you said, you iterate over time on the same defaults that you want to specify in your infrastructure, the infrastructure as a service stuff that you provide to people. If you have a customer that has misconfigured something, or has a less sane default, how aggressively do you want to proactively let them know, or to what extent do you actually have visibility into those less sane defaults they have selected?

[0:22:43.3] MK: Yeah. I think in GKE, it depends on whether you're making the change in the GKE API, or the Kubernetes API. I k now that sounds to a lot of users it's the same. When you're making a change to the GKE API, we do have metrics on how many users overall have turned this on and off? For the Kubernetes API, we look at that a lot less, because it's part of your cluster, that's your data, your configuration. For GKE API, we might know for example how many users have turned on pod security policy, or metadata concealment, network policy. These are good defaults to have.

[0:23:12.5] JM: What is pod security policy?

[0:23:14.6] MK: Pod security policy is a Kubernetes feature that limits the permissions that your pod can run as. If you want to enforce things like set comp for your pods, pod security policy is the way to do it.

[0:23:24.9] JM: What is set com?

[0:23:26.2] MK: Set comp, sorry. Secure computing mode restricts the calls that your application can make to a normal behavior, whatever normal means for your application. It's a feature of the Linux kernel. It's not specific to Kubernetes.

[0:23:38.6] JM: Okay. Tell me more about some of this. If I am using – not to turn this into a promo or something, but we are at Google Cloud; if I'm using Google Kubernetes Engine, what security features am I getting out of the box? Tell me from a product manager, product design standpoint.

[0:23:58.7] MK: Yeah. I mean, you're getting all of Google's underlying infrastructure security. Things like encryption at rest by default, encryption to the front end, in-transit by default and internally when it crosses physical boundaries, Google's custom-built hardware, Google's secure boot and integrity measurement, all the underlying pieces of Google's infra.

Then in GKE itself, Google manages the control plane. It manages the master VMs, the API server at CD all of that for you. We have some documentation that explains how that's secured and there's actually a talk that I gave yesterday that explains a bit more how these components are secured. Then the nodes, Google manages the default configuration, provides patches for your worker nodes and you're responsible for updating those nodes when your patches are available. Then you're still responsible for managing your own workloads.

When it comes to things like specific configurations, like you're asking, what do I get out of the box by default? It's a lot of the hardening of the control plane and a lot of the good defaults for nodes. I mentioned earlier, things are back on by default; things like the dashboard not deployed by default. Basic Auth is disabled by default, client certificates are also disabled by default. The GCE metadata API, the legacy one is disabled by default. Cloud audit logging has a good policy on by default and logs automatically stackdriver.

There's a bunch of stuff that you don't have to configure related to the control plane. Then for your nodes, we put you on cost by default container optimized OS. You can also choose Ubuntu. Cost has a lot of nice security properties. It has firewalls for what port can be used. It has a read-only root filesystem, which is actually what prevented cost from being affected by the

recent RunC vulnerability. It has a secure boot. It's able to be rebuilt reproducibly by Google, since it's based off of Chromium, which means that we can really easily patch it in a case of a security incident, more so than a third-party OS. That defaults, where you don't actually make that choice and that you just get it.

Oh, I'll mention one more actually, because I'm excited about this one. A node auto-upgrade is on by default for clusters created in the UI and is on by default for G-Cloud beta starting in 113. Node auto-upgrade means that we can apply patches to your nodes or worker nodes automatically. Specifically, what makes me as a security person excited is we can apply security patches to your nodes automatically. In some cases where we can apply those patches before the embargo is lifted ,we do. That's actually created both joy and confusion in some of our customers who e-mail me and ask, “There's a new vulnerability. I'm affected. What do I do?” I’m like, “You’re already patched.” They're like, “No, no. It's new. It's from today.” I go back to them like, “No, no, no. You're already patched. How do I explain this to you? You're fine.” Yeah. I think node auto-upgrade is one of the coolest things that we have and rolling it up by default is incrementally in our in our fleet.

[SPONSOR MESSAGE]

[0:26:50.5] JM: When a rider calls a car using a ride sharing service, there are hundreds of back-end services involved in fulfilling that request. Distributed tracing allows the developers at the ride-sharing company to see how requests travel through all the stages of the network. From the front-end layer to the application middleware to the backend core data services, distributed tracing can be used to understand how long a complex request is taking at each of these stages, so the developers can debug their complex application and improve performance issues.

LightStep is a company built around distributed tracing and modern observability. LightStep answers questions and diagnoses anomalies in mobile applications, monoliths and microservices. At lightstep.com/sedaily, you can get started with LightStep tracing and get a free t-shirt. This comfortable well-fitting t-shirt says, “Distributed tracing is fun,” which is a quote that you may find yourself saying once you are improving the latency of your multi-service requests.

LightStep allows you to analyze every transaction that your users engage in. You can measure performance where it matters and you can find the root cause of your problems. LightStep was founded by Ben Sigelman, who was a previous guest on Software Engineering Daily. In that show, he talked about his early development of distributed tracing at Google. I recommend going back and giving that episode a listen if you haven't heard it.

If you want to try distributed tracing for free, you can use LightStep and get a free t-shirt. Go to lightstep.com/sedaily. Companies such as Lyft, Twilio and github all use LightStep to observe their systems and improve their product quality.

Thanks to LightStep for being a sponsor of Software Engineering Daily. You can support the show by going to lightstep.com/sedaily.

[INTERVIEW CONTINUED]

[0:29:02.1] JM: You mentioned securing etcd, just as an example for people who have not heard a previous episode about container security, why is securing etcd so important? What would the vulnerabilities be if I had an insecure etcd?

[0:29:16.7] MK: Etcd is the hard drive for your cluster. It maintains state for everything that's going on in your cluster. It would be like somebody taking your laptop and stealing the hard drive; they get all of your data stored at rest, all of your configurations, they can reproduce your cluster effectively somewhere else and make it look just like your existing cluster, right? It's stealing all of your IP related to your cluster. Also stored in etcd are your secrets. Any secrets that your cluster needs to access, like API, 0Auth tokens, etc., are stored in etcd. That's particularly sensitive information.

[0:29:47.4] JM: What goes into securing that?

[0:29:49.1] MK: Etcd running on GKE is encrypted at rest. We have limited ports that are open, so that it can communicate only with the API server and with other instances of etcd. There's also a feature that we have called application layer secrets encryption to further encrypt the secrets that you have in there at the application layer if you want to. In general, if I'm going to

run etcd myself, it's one of those things that I would want to really, really lock down and not put on the public web.

If I go look on [inaudible 0:30:13.4] today, yes, I can find lots of Kubernetes dashboards, but I can also find a lot of etcd running out there. That concerns me a lot, because I don't think people realize that's a potential attack vector to their cluster.

[0:30:23.9] JM: What would that – like if you wanted to scare somebody straight, what should they be concerned about?

[0:30:30.1] MK: If somebody can access etcd, they can take down your cluster, replicate your cluster. This is assuming you're running anything important in Kubernetes that's dangerous.

[0:30:40.7] JM: Yeah. Does etcd in practice, does it get abused? Do people use it more as this general key value store they can write and read from, rather than just for very narrow purposes?

[0:30:53.5] MK: Yeah, it's a general database. It's not specific to Kubernetes. It just happens to be used in Kubernetes as the maintainer of state.

[0:31:01.3] JM: Interesting. What was your presentation yesterday about?

[0:31:04.7] MK: Yeah. Yesterday I talked about the shared responsibility model in GKE. If you think about shared responsibility, you might have heard this term before. It describes when you're in the cloud, what is the cloud provider's responsibility to secure, versus what is the user's responsibility to secure?

If I look at something like GKE, it falls in that pasi-middle where it's not super clear sometimes what's the user responsibility, versus what's Google's responsibility. It's also not always transparent to the user what Google is already doing to secure them. Trying to better explain that.

[0:31:34.5] JM: Okay, explain it to me.

[0:31:37.3] MK: Google is responsible for the control plane and providing strong defaults for the nodes, including node configurations and the OSS and patches. These are responsible for updating the nodes by applying those patches, unless they have node auto-upgrade turned on, in which case, that becomes Google's responsibility as well. The users are responsible for their own workloads running on top of GKE, just like they'd be responsible for their workloads running on top of App Engine and running on top of GCE.

[0:32:02.1] JM: Go a little deeper into that. What are some of the – as the product manager, I assume you're the product manager around this. Well, shared responsibility, I guess it's not really a product. That's more just a feature of designing cloud services. What are some of the subjective decisions you have to make as a cloud provider when the reality is that it's a shared responsibility system.

[0:32:24.3] MK: The hardest decision to make is how much control you give the user. You want them to be able to do everything they want to do for their workload without shooting themselves in the foot. This is a little bit of a conversation we're having around defaults. If we can turn node on auto-upgrade on by default everywhere, if we can get more and more users using node auto-upgrade, we can patch you in case there's an incident and that's a better – ain't that better for you and isn't that better for us, because we know that you're running a relatively recent version, right?

If I look at something, like SSH into your nods, a lot of users still SSH into their nodes on GKE. That is a potential surface of attack that has a – there's a lot less that we can do to actually control what goes on there. We can initially control if it's user A or user B from your company, if you don't have your own way of controlling how that's managed.

[0:33:09.4] JM: Okay. One subject that I think would probably fall under the category of the user's responsibility is the security policy management, so like who has access – within my organization, who has access to what database, which service can talk to another service? This has been an area that has surprised me with how interesting it is. I guess, the classic way of managing these kinds of policies is you have this big, central security policy manager thing. Every time your service needs to figure out, like can I talk to this other database over here? I

have to go to the central broker of information about my security policy. That can be an expensive network lookup.

Then with the service proxy, or the service mesh model, you have this convenient little sidecar container that does stuff for you. You can put the necessary security policies for your service in that sidecar and the sidecar aggressively update its cache and whatnot. Tell me about that area of the security policy management, particularly the sidecar model as it relates to that.

[0:34:21.9] MK: Yeah. What you're describing is very similar to what we do internally at Google and what exists is being built into Istio. The idea that services service communication shouldn't meet a certain bar, right? Services should be mutually untrusted. I don't necessarily know every service that I've deployed to my environment.

[0:34:39.1] JM: Zero trust.

[0:34:39.9] MK: Yes, exactly. Services should have to authenticate to each other as a result. Services service communication should be encrypted by default. Those kinds of requirements that if I was going to implement as a third-party service, that every other service had to talk to, would actually have been effectively impossible to implement, right? I'd have to have everybody go change the application code and change what they're doing, to then go talk to some central credential service to get a credential, to go talk to the other service. That would never roll out effectively and I'd have to have – the central credential service would have to have an immense amount of load to people to handle all – to be able to handle all the load that was coming in from the other services.

The idea of having a service mesh is that you can have the sidecars and this policy live in between all of these services and enforce that dynamically, rather than having to go through a central service for policy enforcement.

[0:35:30.8] JM: Does the sidecar model, does it materially reduce latency and does that matter for most people relative to the centralized policy manager?

[0:35:40.9] MK: I think it has minimal performance impact. Relative to the centralized policy manager, it's not – I don't know that performance would be what I would be comparing it to. I'll be comparing it to, I don't know, ease of implementation like I already mentioned. The issue is not about – I can still have a central policy. I can still query everything and get my policy centrally somewhere and see what's going on. The issue is more about getting it actually rolled out in my infrastructure and making changes easily when I need to.

[0:36:07.4] JM: To wrap-up, I'm starting to get ready for the next KubeCon, the one in Spain. What are the areas I should look at within the container security space that I may not be looking at right now? Or what projects in the Kubernetes space would you recommend me going deeper into?

[0:36:26.1] MK: Yeah. I think there's a couple things going on. One is this policy, or you just mentioned OPA. If you've looked it up, the open policy agent. There's a lot of excitement and interest around how to use that type of functionality to enforce policy in my clusters, especially when my cluster is living in multiple different clouds, or I have some clusters on-prem and some in the cloud. That's one area of interest.

I think another area of interest, or where I hope to see more things happening is in the container threat space. There's still not a lot of open source tooling to help me figure out what's actually going on in my environment. I really wish some people would start to develop open source tooling there to help to better inform users.

[0:37:01.3] JM: Introspect or something?

[0:37:02.5] MK: To introspect, to detect threats, etc. There's still a fairly vendor-focused –

[0:37:07.1] JM: Vendor specific. Yeah.

[0:37:08.4] MK: Yeah, vendor-focused space. Yeah. That would be an area that I'd be interested in. A related area that that would be forensics, I think as we're going to start to see the first attacks that are targeting Kubernetes and Kubernetes workloads, rather than just targeting credentials like we've seen so far, we're going to have to do snapshots of containers,

which is not something that is well understood. That might be another area. There'll be some other innovation hopefully coming soon.

[0:37:29.4] JM: That's a hard problem, the snapshot problem. It seems hard from – I've talked to one – That's another vendor specific thing right now.

[0:37:37.1] MK: Yeah. Then another area that I've seen a couple of talks on the KubeCon agenda for are around confidential computing, which should be pretty exciting. Confidential computing is the idea that the underlying compute provider, so whether it's a cloud provider, or anybody else who has access to your hypervisor effectively, can't see what's going on what you're actually doing in the VM or in the container.

[0:37:57.4] JM: Is that the homomorphic encryption thing, or is that different?

[0:38:00.6] MK: Homomorphic encryption attempts to solve the same problem, but confidential compute doesn't necessarily have to use homomorphic encryption. It can be done using specialized hardware. There are other methods as well.

[0:38:10.2] JM: Cool. Well, that should keep me occupied. Thank you, Maya. Thanks for coming back.

[0:38:14.9] MK: Of course. Thank you for having me.

[END OF INTERVIEW]

gocd.org/sedaily and learn how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail about building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continuous delivery as much as almost anybody in the industry.

It's great to always see continued progress on GoCD with new features, like Kubernetes integrations, so you know that you're investing in a continuous delivery tool that is built for the long-term. You can check it out for yourself at gocd.org/sedaily.

[END]