EPISODE 592

[INTRODUCTION]

[0:00:00.3] JM: Deploying software to a container presents a different security model than deploying an application to a virtual machine. There's a smaller attack surface per container than per VM, but the container is co-located on a node with other containers, so there is what's known as a breakout risk of somebody getting outside of the container that they're supposed to be allocated to and getting into another container on the node.

Containers are meant to have a shorter lifetime than VMs, so there are potentially fewer consequences if a container needs to be destroyed and rebuilt due to a potential security vulnerability. There are many other trade-offs between the container model, versus the VM model, but generally it looks containers are better for their security properties.

Maya Kaczorowski works on container security at Google. In a recent talk at KubeCon, Maya discussed runtime security of containers on Kubernetes. Maya joins the show to discuss container security and what it means to software developers and operators. Maya also gives guidelines for evaluating the security of your own cluster. We talked about the security benefits of a managed Kubernetes provider and also explored how some container security vendor software works.

We touched on Istio and also at the end, talked a little bit about Chromebooks, which have their own interesting security properties. I enjoyed having Maya on the show and I hope you enjoy the episode as much as I did.

[SPONSOR MESSAGE]

[0:01:45.6] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with more than 200 technologies, including Kubernetes and Docker, so you can monitor your entire container cluster in one place. Datadog's new live container view provides insights into your container's health, resource consumption and deployment in real-time.

Filter to a specific Docker image, or drill down by Kubernetes service to get fine-grained visibility into your container infrastructure. Start monitoring your container workload today with a 14-day free trial and Datadog will send you a free t-shirt. Go to softwareengineeringdaily.com/datadog to try it out. That's softwareengineeringdaily.com/datadog to try it out and get a free t-shirt.

Thank you Datadog.

[INTERVIEW]

[0:02:42.0] JM: Maya Kaczorowski is a security product manager at Google. Maya, welcome to Software Engineering Daily.

[0:02:46.8] MK: Thanks for having me.

[0:02:49.1] JM: You work on container security for much of your job. If I'm an average developer, why do I need to care about container security?

[0:02:57.4] MK: Well, if you're an average developer, you're actually probably not using containers yet, so maybe you don't need to carry yet, but it is a trendy topic these days. You see more and more workloads moving to containers and more and more workloads worrying about container orchestration, jar systems like Kubernetes, for example.

If you are running something in a containerized environment, then you should care about the security with that, and so that's container security. In terms of should you care more than security of VMs, or security of other infrastructure systems? Not necessarily. We're not necessarily seeing attacks yet that are targeting containers specifically, but it's just a good general best practice.

[0:03:31.8] JM: In comparison to virtual machines, what are the security advantages of containers?

[0:03:38.6] MK: Sure. A container has a bit of different construct than a VM. A VM will have hardware with host OS running on top of that, a hypervisor which helps isolate individual VMs and then the VMs running on top, each with their own binaries, libraries, application code, etc. A container on the other hand will have no hardware and then a host OS. The host OS will be much smaller. It's a minimal host OS is what we say, and you won't have a hypervisor. Then you'll have a container runtime and shared binaries and libraries and then individual containers with application codes running on top of that.

The major difference is from an architectural point of view, are that containers have this minimal host OS, rather than a bloated host OS with all the stuff in it, which is good because it has a smaller surface of attack, than a VM would have in that host OS. The downside from a container point of view is that it doesn't have this hypervisor, and a hypervisor is something that we're very familiar with in VMs, in terms of the isolation that they provide for workloads. We don't have this similar construct, or we didn't until the last couple of months have this similar construct for containers in terms of isolation.

[0:04:41.2] JM: The container has a small attack surface. What does that mean to have a small attack surface?

[0:04:48.3] MK: It means that it has a fewer components that are not needed. An application that's very large will have a large attack surface, but what a container gives you versus say, a VM is fewer unnecessary components. A VM might come with for example, your post image might have various language libraries and packages and things in that that the actual end-application that you're using doesn't need. From an ease of management point of view, it's easier for you to just have the same VM image for all your VMs.

With containers, you can have a same container image for – a container host image for all your containers and then put a lot of that bloat into individual containers, so you don't have to have certain services have packages for other services if they don't need them. Why does that matter from a security point of view, other than obviously it helps your performance, is that say there's a vulnerability discovered in Java next week, right? Or discovered in a particular kernel, piece of the kernel, or discovered in a particular application, whatever.

With VMs, you'd have to go through everything and see what's affected. With containers, it might actually only be some fraction of your infrastructure that's affected. More importantly, you won't have things that don't even use that component being affected by a security issue.

[0:05:52.3] JM: The best practice around containers is to treat them as cattle, not pets. That's the metaphor we're all familiar with at this point. The idea is that cattle would be more dispensable. How does the metaphor there, the idea that your containers are going to be more dispensable than VMs, that there's – it's okay if your container dies. How does that impact the security properties of containers in terms of how we actually use them?

[0:06:21.9] MK: Yeah, so containers are meant to be exactly that immutable, and therefore often come up and down, right? They're supposed to be generic and then have this image that you have to redeploy every time you want to do something, rather than you tweaking what's happening already in production.

What that means is that your container has a shorter lifetime than a VM would have, which is a good thing from a security point of view, right? It's harder for somebody to gain a foothold into a system that's constantly changing, and where that container constantly gets killed and restarted. The downside from a container security point of view is that if you do have an attack, the container might be gone by the time that you figured out there's an attack and you might have, and no way of doing forensics. It's great that you're your environment is constantly changing, but it doesn't necessarily let you introspect individual events happening in your environment.

[0:07:02.8] JM: Right. Is there any risk there of auditability, because if my containers are churning out more often, then somebody could potentially break into my container and then do to a routine event, and then they break into my container, they do something with it, and then the container goes away, because of some churn and the infrastructure that is to be expected, that's totally fine. If that thing happens, how will I ever be aware that somebody has intruded and has made their way into my container?

[0:07:34.3] MK: Yeah, you might not even know. I mean, the idea is that you definitely have – you should have logs for what's happening to your infrastructure and be able to look at it that way. What you won't have is deep forensic analysis. You won't be able to a snapshot your

container, like you can a VM today to see what changed between this moment and that moment. We don't have the same level of forensics technology for containers yet.

That being said, the scenario that you're talking about if somebody getting in and seeing that they're in a container and doing very specific behavior, because they're in a container is extremely rare, right? People are attacking containers the same way that they're already attacking VMs. They're saying, “Hey, free compute resources. Let me mine some cryptocurrency.” It's not like, “Hey, I'm in a container. Let me try to take this person's service down by crashing the container repeatedly. Let me try to flood the event pipeline by making sure that we can't detect what I'm actually going to do next, because it's crashed.” That's not what we're seeing yet.

[0:08:22.8] JM: If I'm running my container on the same host as some other container that's totally unrelated to what I'm doing, how well is my container isolated from that other container?

[0:08:33.7] MK: Yeah, that's the hypervisor security question. On a VM, your VM is pretty well isolated from other VMs using hypervisor technology. From a container point of view, a container isn't really meant to contain. You have containers running in the Kubernetes framework, you have containers running in pods and pods running on nodes, and nodes have that hypervisor layer isolation that we have.

Two things that are running in the same nodes of two containers in the same pod, or two pods in the same node don't have a strong security boundary between them, traditionally. There's a couple projects in the space right now that are trying to address that. That problem, one is contact containers, which aims to run a very lightweight VM per container, so that you can have benefit from the hypervisor area isolation basically work only with containers.

The other project is G visor, which is released from Google, and something we've developed internally, I've been using for years, and that's around providing a fake kernel and user space, such that you can have two containers running on the same machine, which don't have access to a shared kernel.

[0:09:32.7] JM: We did show a while ago about the fact that companies often have infrastructure sprawl. They don't know what's running across their company. They just have so much infrastructure and it seems with containers, it becomes even cheaper to spin up infrastructure. In fact, you and I were both at KubeCon, and a lot of the thrust behind some of the enterprises that we're walking around KubeCon, talking to different vendors and deciding which vendors should I go with, why should I go with the Kubernetes vendor?

I think a lot of the thrust was the idea that if you install Kubernetes in your company, it becomes much easier to spin up infrastructure. You can do testing, you can spin up staging environments. There's a lower barrier to doing that. The consequence of that is that you just get more infrastructure, and it seems like you could get orphaned to containers quite easily. Is that a problem that increased infrastructure sprawl?

[0:10:30.9] MK: I don't think so. It's not yet, or rather not that I'm seeing. People are typically taking existing workloads and putting them into containers, or building brand new things from containers from scratch, but we're not yet seeing – containers aren't old enough to have legacy infrastructure yet, you know what I mean? There aren't these containers that you build three years ago lying around that are still running.

Is that something that we could definitely have to deal with in a couple of years? Maybe. The idea that I'm liking right now in security is something we're calling reverse up time. It was championed by Docker, who's obviously knows something about containers. The idea is that you shouldn't have any containers in your infrastructure running more than a certain amount of time. They're not providing guidance on this, but you could imagine being able to say, "Hey, I want to only have containers that are less than a month old, or less than a week old, or less than a day old in my infrastructure." I'm going to constantly kill the containers that are too old and rebuild and redeploy them.

Now, the reality is I don't actually know of anybody doing this in practice. Google deploy so many containers, we don't purposely killed the old ones like this, but we deploy so many containers that it naturally has a turnover that's quite high. The other end is some companies will rebuild their containers every night. There was an example of that at KubeCon. I believe it was the Financial Times that was saying, "We deploy every night. We rebuild every night," so

that if you need to redeploy in the morning, you can, but nobody's doing that full flow yet of saying, “Here's what I've already deployed, looking at my deployment history. Here's how old it is. Therefore, let me kill it, or let me determine that it's part of this rogue too old infrastructure that I need to deal with, and then let me rebuild and redeploy.” That trigger, that cycle doesn't exist yet as a construct that people are following.

[0:12:02.3] JM: What do you think of that part of that the Financial Times has there, where on a nightly basis, you just scrap everything and rebuild it?
[0:12:09.9] MK: I think it's great. I wish other people would think the same way. Computers are meant to be immutable and were meant to be constantly redeploying them, and your infrastructure is meant to be constantly changing. I think a lot of users I'm seeing are saying, “Hey, I want to do the Kubernetes thing,” but they're not grasping that concept. They're deploying the Kubernetes and deploying version 1.8, and then they freak out when they have to upgrade to version 1.9. That's not the point. The point is that you've changed to an infrastructure that is constantly changing, where it's so much easier for you to patch, so much easier for you to upgrade.

[0:12:39.6] JM: Right. Let's say I'm a CIO at a company and I get to a situation where I have tons of containers across my company, is there a good way to be able to index and identify and introspect all of these different containers and perhaps maintain some security hygiene across this sprawling infrastructure?

[0:13:01.4] MK: Yeah, we're looking at container security in three aspects from a Google point of view. The first one is what we'll call secure to develop, which is what you do in order to properly develop containers from your infrastructure, things like networking, identity, secret management, etc. There's a set of best practices to follow there. I know the Kubernetes community right now is discussing how to make better secure defaults for the average user to be able to deploy something in Kubernetes that's pretty secure. That's step one.

Step one is use the things that are already there. Don't make life harder for yourself than it needs to be. Step two is the, what we're calling secure to build and deploy, or the software supply chain, which is how do you give your developers good images, so that they can then build containers on top of those, have a hermetic environment to build them in, scan those

things for vulnerabilities, ensure that they came out of your CICD pipeline before you deploy them and enforce other deploy sign policies.

I think there's a lot of emphasis being put right now in the industry on that, and that makes a lot of sense, because it's a single choke point for what actually ends up in your infrastructure. If you're not looking at what some minimum requirements are right now for deploying things for infrastructure, that might be a really good first step. That's also how you're going to get a deployment history of here's what was deployed at my infrastructure.

The last area is what we're calling secure to run, or runtime security, which is I already have a container up and running in my infrastructure and I want to be able to detect an event that's bad, that's happening, monitor that container, move it to a different network if that's an option, kill it, restart it, whatever it happens to be. Then after the fact, do forensics. That's the third area.

To answer your earlier question, if you're thinking about this and how do I know what's happening in my infrastructure right now? Is there a good way to introspect? I would say not really. I think the best thing you can do today is start with some secure defaults and then monitor everything that you are deploying to see – so that you at least have an idea of what could be in your environment. You don't know that it's died or not, or if it's still in your environment, but you know that at some point, you deploy that it's your environment.

[SPONSOR MESSAGE]

[0:14:57.4] JM: Software workflows are different at every company. Product development, design and engineering teams each see things differently. These different teams need to collaborate with each other, but they also need to be able to be creative and productive on their own terms.

Airtable allows software teams to design their own unique workflows. Airtable enables the creativity and engineering at companies like Tesla, Slack, Airbnb and Medium. Airtable is hiring creative engineers who believe in the importance of open-ended platforms that empower human creativity.

The mission of Airtable is to give everyone the power to create their own software workflows; from magazine editors building out their own content planning systems, to product managers building feature roadmaps, to managers managing livestock and inventory. Teams at companies like Conde Nast, Airbnb and WeWork can build their own custom database applications with the ease of using a spreadsheet.

If you haven't used Airtable before, try it out. If you have used it, you will understand why it is so popular. I'm sure you have a workflow that would be easier to manage if it were on Airtable. It's easy to get started with Airtable, but as you get more experience with it, you will see how flexible and powerful it is.

Check out jobs at Airtable by going to airtable.com/sedaily. Airtable is a uniquely challenging product to build, and they are looking for creative front-end and back-end engineers to design systems on first principles, like a real-time sync layer, collaborative undo model, formulas engine, visual revision history and more.

On the outside, you'll build user interfaces that are elegant and highly customizable that encourage exploration and that earn the trust of users through intuitive thoughtful interactions. Learn more about Airtable opportunities at airtable.com/sedaily.

Thanks to Airtable for being a new sponsor of Software Engineering Daily and for building an innovative new product that enables all kinds of industries to be more creative.

[INTERVIEW CONTINUED]

[0:17:16.7] JM: Now you mentioned this case of cryptocurrency mining that can occur if somebody accesses a container, or any piece of infrastructure, and they get privileges on it and they can install cryptocurrency mining software. They might do that. In order to prevent that thing from happening, or at least detect what thing when that thing happens, you have to have some infrastructure in place to detect behavior that deviates from the norm.

Now in the cryptocurrency mining case, I can imagine pretty straightforward. You just look at what are your containers is taking up inordinate amount of CPU, probably something's going

strangely. There are other cases where the detection of behavior that is deviating from the norm is perhaps more subtle, like you have accessing private information, for example. How would you detect behavior that would deviate from the norm if somebody has intruded your infrastructure? What's the strategy for being able to detect it on a regular basis?

[0:18:21.6] MK: A lot of the runtime security solutions are looking at system calls, so they typically are deployed as either a kernel module, or a privileged container that you have running on every node, and will look at the system calls that your containers are making to determine if they're doing something unusual. A lot of these have both a set of rules that's detecting common attacks. Plus usually an ML model that would be trained on your specific infrastructure that says, “Hey, we've never seen you make this access before. Therefore, maybe this access should be flat.” That's definitely how you should be detecting things that are happening in your containers.

However, what you're describing, a lot of the cryptocurrency type stuff that we're seeing is way more boring, right? A lot of the conversations I have with customers are like, “Hey, I'm really afraid of a container escape. I'm really afraid of somebody getting in to my containers and escaping and then doing something terrible with my infrastructure, or even I'm worried about somebody getting into my container and mining cryptocurrency.”

The reality is that's not what we're seeing. People are not attacking specifically containers yet. They're seeing opportunities to use those resources. Even things like the Tesla hack, which is I guess semi-famous now in the in the container world, probably because it's one of the few public examples that we have of a hack involving Kubernetes, is that they had the UI dashboard running, the Kubernetes UI dashboard running expose that is without a password, that included their AWS IM credentials.

An attacker found this exposed dashboard, found their credentials and started crypto-mining. They weren't even mining the crypto in the same cluster. They just took the credentials from a thing and then started using them elsewhere. It's not container-specific necessarily yet. It's the same things we see elsewhere; privilege escalation, credential theft, and I use a zero-day vulnerability to then get something valuable, which in this case is typically an IM credential, to then do a cryptocurrency mining.

[0:20:00.2] JM: Although I have seen you – you gave a presentation at KubeCon and you were talking about this in a little more detail and what you got into was container monitoring and logging can potentially assist with this situation.

[0:20:14.3] MK: For sure.

[0:20:14.9] JM: What are some best practices around container monitoring?

[0:20:18.3] MK: I'll just give this for this specific example, right? One thing that we could we could look for would be is a dashboard open, is your Kubernetes UI dashboard open? When was it last accessed? Where we expect to see in a lot of attacks is like, is somebody trying to get a reverse shell? That's very clearly somebody doing a malicious action. They're not yet doing the cryptocurrency mining. They're doing something to try to get something valuable to then do something bad.

In terms of monitoring, to sorry, to answer your question, it's a combination of things. I think if we look at something like Kubernetes, pillar monitoring, application calls and application things using things like Prometheus, we have some network monitoring that's going to come from tools like Istio. We have SIS calls or coming from some of these open source, or third-party container runtime detection tools. We have audit logging built into Kubernetes as well, that tells you who did an action on the master API and when did that happen and what action did they perform.

There's a wide variety of logs that are coming out of this. I think what we don't necessarily have yet as an industry is a good way of saying, “Hey, these three actions here bundled together is probably an attack on a container.”

[0:21:19.7] JM: Yeah, I think you said earlier you could at least train a machine learning model and then be able to detect deviations from that model, but even that's not going to be perfect.

[0:21:29.6] MK: I mean, it's better than nothing.

[0:21:32.3] JM: Certainly better than nothing, yeah. What about that monitoring and logging? I mean, these are things that you would want obviously in a Kubernetes cluster. Are the practices

around monitoring logging standardized enough and are customers deploying them on a reliable basis?

[0:21:51.7] MK: I think if you're working with the hosted solution, so something like a GK Kubernetes engine from Google, a lot of that's deployed for you, so audit logging is there for you by default, Prometheus is there for you by default. Obviously, we're working on making Istio available for you there. If you're doing it on your own, I think it's well understood what you need to do, but it's maybe not it's really easy for you to do yet. There's still a lot of work that we're asking an average developer to take on.

We're talking about monitoring, but that's true in security too, right? We give you a laundry list as a community right now that says, “Oh, you want to be secure in Kubernetes? Sure. Here are 15 things you have to go do.” People are starting realize that that's not going to work, that it's not going to work for the average user and it's not going to work for enterprises and are starting to change that and fix that.

[0:22:31.8] JM: We've done several shows on Istio. This is that service mesh tool. What role does a service mesh play in security policy?

[0:22:41.6] MK: Yeah, a service mesh can see the network traffic between various services that you have, and can therefore both help you monitor the network traffic, as well as enforce arbitrary policies that you might have. You might want to be able to say, “Hey, the service shouldn't ever talk to this database. This front-end service should never talk to this back-end customer database and create that segmentation.” You could also say, “This backend service should talk to that customer database,” and suddenly it started asking for it a large amount of data that it's ever asked for before. Anomaly detection, therefore we think possibly somebody is trying to extract data.

[0:23:13.6] JM: I know this is not what you work on from a day-to-day basis, but what are some other uses of Istio? Why does that project have a lot of steam around it?

[0:23:22.1] MK: I can talk a bit about how it relates to what Google does internally, which is particularly interesting. Internally, we have something called Application Layer Transport

Security, ALTS. There's a white paper on our website. An ALTS provides strong identity for services in our internal infrastructure, and uses the identity to authenticate services and encrypt data in transit when it crosses physical boundaries.

The benefit of Istio, obviously from a service mesh networking point of view is great. The benefit from a security point of view is huge. You have these services who previously you just had to trust that they were whoever they said they were. Now you're going to actually verify that they are in fact the service that they said they were. Getting a request from this database, getting a request from the front-end service for a particular user's data, or whatever.

Within Google is authenticated and verified that that is in fact the service it's requesting for. There's in fact a valid end-user request coming from with that service. With Istio, that's going to give everybody else the same power. Everybody else will be able to say, "Oh, yes. You are in fact that identity." Therefore, I can have this interaction with you. That's huge in terms of limiting access to various calls within your infrastructure.

[0:24:29.5] JM: Now to go back to the logging and monitoring conversation, I think this also fits in with Istio, because if Istio is giving you a bunch of network traffic data, then you've got network activity, you've got process activity, you've got file activity. You have all these sources of high volumes of logging and monitoring data. If I'm operating a container infrastructure, how do I filter through all this information? How do I find signal from the noise? What am I configuring?

[0:25:01.4] MK: Hopefully you're not trying to find signal from the noise hopefully, whatever vendor open source tool has done that for you. You're looking for anything that's unusual behavior. If you train on ML model and say, the last 24 hours of access are normal, tell me anything that goes beyond that access. It will tell you – we'll find all kinds of things for you, That's still pretty noisy.

I think what we don't have yet is necessarily, "Oh, you're deploying Redis on Kubernetes, we know what that looks like. Therefore, this behavior is unusual. Or we don't have these templates of expected behavior for various services yet.

[0:25:32.4] JM: Hopefully some of the vendors do, or somebody has them someday. On each node, the Kubernetes model is there's several pods, several pods fit into a node and their containers within each of these pods. If we're trying to detect malicious behavior, we are going to look at system calls. We need to look at the system calls that the containers on a given node are making. For those unfamiliar with Linux, what is a system call? Can you explain that term?

[0:26:04.5] MK: Sure. A system call is a call that a particular process would make to the kernel to do something like process a particular function. Something like a memory access, something like dumping memory, except this would be reflected in system calls.

[0:26:20.6] JM: Why is it useful to be looking at system calls from a security standpoint?

[0:26:25.1] MK: From a security standpoint, system calls are the one of the main ways that a container would try to do anything. What I mean by that, is if a container can't do something in-memory, in its own application, it needs to ask the kernel for more memory, more resources to connect to another container, to do anything. All of that has to go through the Chrome. A system call is going to be a large fraction of what we see the container doing will be reflected and the system calls on the containers is doing.

[0:26:49.9] JM: In some of the security solutions that people deploy to Kubernetes, you'll have a model where one of the pods on the node is a management container. Explain what a management container is.

[0:27:03.3] MK: Yeah, so a Daemon set, or a published container is a container that you would run on every node and running in privileged mode would have the ability for example, to look at another container system calls. A management container in a typical model, we're talking about a runtime security solution is another container that would look at the output of that privileged container, or that kernel module and say, “Hey, these calls look anomalous.”

When we're talking about having an ML model that you're running on your containers and to look for anomalous behavior, that itself is running in another container, and that's typically running in a management container. Now the reason you would want to run that container alongside all of your other containers is then if you want that container take any action. If you

just want to know that something is happening, you can run that independently and run it somewhere else. If you want to then to be able to say, “Okay, please kill that container.” That management container needs to have the power to do so and take that action.

[0:27:49.6] JM: Okay. How does information get shuttled between the nodes and the management container?

[0:27:55.9] MK: Just like any other information, would go between two pods, or two containers.

[0:28:00.7] JM: Okay. What kinds of data are being forwarded from those nodes and the management container?

[0:28:06.7] MK: From a container that you're just monitoring, you would look at the system calls that it makes to the kernel, and that's what either the kernel module, or a privilege container would see. Then it would feed some subset of that information, some subset of logs to a management container to then take action on, analyze, etc.

[0:28:23.0] JM: Okay. The management container could be reading these feeds of system calls that all the other individual containers are making. If it detects some anomalous stream of behavior, the management container can emit a signal to the bad actor container, or the deviant container and shut it down?

[0:28:41.7] MK: Correct, correct. The only thing I'll add to what you're saying is a management container would probably not be looking at all of the SIS calls. Typically, the privileged container kernel module would filter some subset of things that it thinks are more critical.

[0:28:53.6] JM: Okay. Are there a specific set of system calls that are typically filtered for?

[0:29:00.8] MK: It's just more from a performance point of view. If you're feeding it all of the SIS calls, then that's a lot of overhead. Looking more at things that are more likely to be related to a security attack.

[0:29:10.6] JM: Okay. This is again something that hopefully the vendor, if you're purchasing some security software from a vendor, the vendor is going to help you set up the management container and the instrument, the daemon set across all your other containers, and they're going to know what to look for essentially, or they'll purport to.

If I find deviant behavior, what is the effective response strategy? You've got a bunch of options. As you said, you could just kill the container, but you could also send an alert, you could isolate the container, you could pause the container, you could just restart it. What are some of the use cases where you'd want to take these different responses?

[0:29:49.3] MK: Yeah. I think it really depends on how bad it really is in your infrastructure. If you're dealing with – you just have a security vulnerability, or something like that, the easiest thing to do and it's a known zero-day, the easiest thing for you to do is just to rebuild, redeploy your container and that's fine.

If you're talking about actually being attacked and trying to take an action like, “Oh, something bad is happening in my environment.” The first thing you can do is just send an alert, right? This is a easy thing to just let your team know, let your security team know that something bad is happening, to go take a look if they have capacity and then start to investigate.

I typically recommend, this is typical security best practice is before you start implementing any automated reaction, just start sending alerts on things and seeing what comes up and seeing if they’re legitimate. Then you'll have a good idea of whether or not you want to take the next step, which is trying to automate some of these reactions.

The simplest reaction you could probably take is to isolate a container. What I mean by isolate is you can move the container to a new network. That would then restrict its network connectivity, the idea here being, I don't want – if there is something bad in this container, I don't want it to spread to other containers. I don't want it to get to other parts of my network. I also want to still have this container around, so I can look at it a little bit. This goes back to the forensics piece that we had earlier that you might not have a great idea of what's actually happening in that container, and so killing it is going to get rid of that information. If you move it somewhere else,

you can still look at it later if you have a chance to. At least, it can't affect certain parts of your infrastructure that you were worried about.

The next one would be pausing your container. You can suspend all the running processes in the container. For example, if you detect a crypto mining, this would be a great way to somebody from mining Bitcoin all night long, and then you can get backup of that again to do forensics and further investigate. That's another in the same order of magnitude of trying to stop it from getting worse, but not necessarily stop it completely.

Then there's the most – the strongest actions you could take, which are restarting a container, or killing a container. Restarting a container is a great way for you to just stop whatever is happening right now from happening, that's why we do health checks on containers, that's why people constantly rebuild, redeploy, etc.; just killing the thing and restarting it.

The only problem with that is if somebody was able to already get a foothold into your container, they'll lose that, but it doesn't solve the problem and it doesn't stop them from doing it exactly again, exactly the same way again. I would say you're going to restart a container if it's a really critical threat to your infrastructure and you have a pretty good idea as to how it happened.

Or else, you're just going to end up in the same situation again in 10 minutes from now. The last say here would be, killing a container, you can just halt all running processes and not even relaunch the container. This is the absolute worst case scenario that you can be.

[SPONSOR MESSAGE]

[0:32:17.0] JM: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more, people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A 15-dollar flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect

amount of resources for your application. There are also CPU-optimized droplets perfect for highly active frontend servers, or CICD workloads.

Running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with.

You can make a $100 go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure and that includes load balancers, object storage, DigitalOcean spaces is a great new product that provides object storage, and of course computation. Get your free $100 credit at do.co/sedaily.

Thanks to DigitalOcean for being a sponsor. The co-founder of digital ocean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[0:34:24.0] JM: Much of the importance of the security infrastructure today in 2018, it sounds like is what the commercial provider of security is offering you, for example, in that management container. How did the open source solutions that are available for container security, how did those compare to the commercial providers?

[0:34:46.2] MK: I think the open source community is really focused on going back to our earlier secure to develop, secure to build the deploy, secure to run. The open source community is really focused on the secure to develop. There's a lot of efforts going on right now to improve the defaults and Kubernetes, to improve the defaults and that customers get to use and some of these other tools, to make it easier to use these things.

There's not that much open source attention being put on the second and third item, not because they're not interesting, just because the community hasn't gotten there yet, I'll say. In the middle space in terms of software supply chain, Google has an open source project called

Grafea, which is a metadata server, which lets you for example, store information about vulnerabilities and your images.

You would scan your images for vulnerabilities, or have other requirements around the images and you could write these to this open source metadata server. That's what we use for our container scanning capabilities. The last area in terms of the runtime space, the pure detect something bad that's going on, you were correct, there's really not a lot in the open source right now.

From the handful of companies that are involved in this space, there's a couple of projects; one is Sysdig and Falco from Sysdig. It's a kernel module and a rules engine that again, look for anomalies and whatnot in your environment, and they plug into their other commercial solution. Another one is more in the networking space, which we haven't really talked about is Cilium from a company called Covalent. It does some network traffic monitoring and isolation type stuff. Another one in the space is Capsulate. There's another company in the space and they've open sourced, again they're – I believe it's a privileged container that they deploy.

[0:36:13.7] JM: Until let's say, I'm an enterprise and I'm walking through the expo hall at KubeCon and I'm talking to the different security providers of which there are many, what are the questions I should be asking them?

[0:36:26.7] MK: I think you should be realistic first, which is have you what you need to get done in your own environment before you go to a vendor? Anybody who can help you with basic configuration deployment stuff, one example is the CIS benchmarks, or if you have a compliance needs around PCI, for example. Any vendor who can help you set up your cluster the right way, that's probably where I would start, in terms of getting the basics down right. Once you figure that out, then let's look at vulnerability scanning and image scanning, that kind of thing.

Can you help me figure out how to scan, to only deploy things that I've scanned for vulnerabilities and have no known vulnerabilities? That's a pretty big ask. Then I would look into the runtime space and what you can do there to detect what's happening in your environment.

It's not that the runtime space is less important, it's as important. I'd say it's less urgent, less low-hanging fruit. People aren't there yet.

[0:37:13.7] JM: Are there any security anti-patterns that you see commonly executed with people deploying Kubernetes infrastructure?

[0:37:21.4] MK: I think it goes back to this defaults thing. It's very interesting to me. I only started working in Kubernetes about six months ago. It's interesting to be coming in as a outside security person. Seeing a lot of things that I would have done “differently,” or would have expected to be different, one I'll look back is our back was only introducing Kubernetes until 16 and made the default in 18. Our back is role-based to access control.

It blows my mind that we got a year and a half of Kubernetes releases solely under our belts without having our back. I think that completely blows my mind. The anti-pattern that I have there is that people are still running cost-reducing a back, or clusters in 16, or in 17 that could be upgraded to our back and aren't upgraded yet.

The more general anti-pattern that I have there is what I was talking about earlier, which is people expecting their infrastructure to be constant and non-changing, right? Part of the goal of moving to microservices, to containers, to all these things, is that you are constantly rebuilding and redeploying. One of the things you'll be doing is constantly upgrading. You'll be constantly upgrading Kubernetes and get all the new features.

Being in a mindset, you've tied yourself to a particular feature, or a particular version becomes a problem from a security point of view, because you want to be able to patch that thing as soon as it's available.

[0:38:31.4] JM: What security stuff were you working on before you move to Kubernetes?

[0:38:34.0] MK: Yeah, I was working on encryption at rest and encryption key management for Google Clouds. I worked on Google Cloud KMS and a couple of white papers on how we encrypted data at rest and in transit by default.

[0:38:45.0] JM: Fascinating. How did that work change your perspective on security?

[0:38:48.9] MK: That previous work?

[0:38:49.8] JM: Yes.

[0:38:50.4] MK: I don't know that it changed my – Well, I think it changed what I see customers ask for, in terms of security, right? Everybody wants to be secure, everybody wants some minimum requirements. Google is great and that Google Cloud will encrypt all your customer content by default without action required from you the customer, which is amazing. Yet, customers really want the control of managing their own keys.

If you look at something like key management, encryption is not hard, because encryption is hard. I mean, it is, but encryption is really hard, because key management is so hard. We've done this wonderful thing, in my opinion, just manage the keys for you and made it really easy for you. A lot of enterprise customers still want to manage their own keys. To me, it's like, you want the worst of it. You want the brunt of this thing, with almost no benefit, because the benefit is already there. At the same time, I completely understand, right? Enterprise requirements, that's what they're going to ask.

[0:39:41.5] JM: Well, you see the same thing in, not to go down this rabbit hole at all, but the cryptocurrency community. Like you have the people who just want to buy cryptocurrencies on Coinbase, which does the key management for you, and then you have a ton of people who are saying, “No, that defeats the purpose. I want to maintain my own keys.”

[0:39:59.3] MK: Yeah. I mean, it's a shocker that we don't have a working PGP e-mail solution, but that's a whole other forms.

[0:40:06.8] JM: Yeah. What else is Google building to help with container security? Are there any problems in the container security space that we haven't addressed in this conversation?

[0:40:17.0] MK: I think we've touched all the major parts of it. I'd say there's definitely some very interesting work being done in isolation, and that's what I was referring to earlier with G

Visor, and we're involved in that in the Kubernetes community in terms of making – what does a sandbox pod look like and what property should it have, and how do we actually protect it from touching things, like shared storage and shared networking, and all these other things that are also shared that are not the kernel, right?

Another area that we're obviously spending a lot of effort on is this software supply chain piece, so Grafea is the metadata, open source meta data server that we spent quite a bit of time working on. How should that work as part of a standard flow? Is there something that community can do to make that even easier for a developer not to mess up and have it flow through from your standard CICD pipeline through production?

Then the area that I'm primarily working on is this runtime space, where we don't have anything to announce it yet, but obviously working to make it easier for you to run partner tools on GCP, and for you to have some base protection in place that you would need, if you're running containers.

[0:41:11.8] JM: That Grafeas metadata project, tell me more about how that assists with security?

[0:41:18.4] MK: Yeah, so the idea is you might have a container registry somewhere of all your images, and you want to be able to record information about those images, such as when was the last time they were built? Does this image meet my internal PCI checklist? Does it have any known vulnerabilities? Can I deploy it in this environment? All those kinds of things. Various metadata about that image, and one of those pieces of metadata is vulnerability information, and one of those is what we'll call attestation information, which is I attest that this thing can be deployed, or attest that this thing cannot be deployed based on for example whether it has certain vulnerabilities.

The Grafeas metadata project is really around creating a standard for what that looks like and a easy way for you to run your own metadata server for your own image registry, such that you can make those decisions based on properties of your images. You can build an admission control over that, then says if this particular image does not meet that vulnerability mark, or that attestation mark, do not deploy.

[0:42:12.6] JM: Right. Yeah, that sounds quite useful. I think we covered that a little bit in a previous episode with IBM, because that's – IBM's working on that as well, right?

[0:42:22.2] MK: There's a handful of partners working on JFrog, Aqua, Twistlock.

[0:42:27.7] JM: Okay. Last thing I wanted to ask you about, you use a Chromebook and I have just heard about –

[0:42:33.7] MK: How did you know that?

[0:42:35.7] JM: You hear about people, like you hear about entire enterprises switching over to Chromebook, because of these security advantages. What is the advantage of the Chromebook security model?

[0:42:45.9] MK: Yeah, the Chromebook security model follows the Google security best practices in the sunset. First of all, to understand why Chromebooks are so popular with a Google, it's important to understand how we work. We don't have code locally on our machines. We do all our code builds and development in the cloud. Your laptop, if you have one, is basically a browser. It's a way for you to get to your code editor, or SSH into a machine, or do that thing. Chromebook make a lot of sense for Googlers internally to use for development.

The security model is that it has – a piece of hardware has a TPM that's trusted that we can use to bootstrap identity to. The machine has an actual identity; it has a machine certificate that we that we invented that in that TPM. That's then used as part of – as one of, I should say many risk factors that we use as part of beyond corp, so determining whether or not your access to a particular application should be allowed based on for example, your location, your ID, your password and the machine certificate that you're accessing from, etc. Chromebook gives us that.

The other big advantage for us from a Chromebook is that we control the bills. We know what Chrome OS looks like. We could rebuild Chrome OS from scratch. It's open source project, anybody can take a look at it. Having that level of control over the OS and building in a ton of

security features from the get-go, like how load pin is using that stuff gives us far more control over knowing what exactly is running on our machines.

I think the benefits for a general individual versus a Google individual are slightly different. I think Google has a lot of Google specific things that make sense for Chromebooks. As a general individual, why would give it to my mom, would be it's significantly harder for her to be phished, given that you're living in an ecosystem around having your Gmail account already has a ton of protection around it, you have a hardware token in the machine, it's rebuilt, it's repatched for you automatically, all that stuff.

[0:44:29.2] JM: Okay. Yeah, I had a listener that wrote in to me recently about why he was so excited about Chromebooks in the future, and just telling me about how he saw a world where every application was just ephemeral containerized infrastructure. There would be so many advantages to having every application being ephemeral containerized infrastructure, that you really wouldn't want any of that application activity occurring on your client device, unless maybe that client device is running a containerized system itself. I haven't fully delved into that paradigm, but –

[0:45:06.9] MK: There is a project in that space, an open source project called QUBES, and their goal is to make everything you're doing on a local device in a container. It's a super cool project. It makes a lot of sense.

[0:45:20.0] JM: Yeah, sure. Why not? I mean, all the security advantages that we've been talking about for this entire episode, why wouldn't you want those on your client applications?

[0:45:29.4] MK: Yeah, good question.

[0:45:31.9] JM: Okay. Well, Maya thanks for coming to the show. It's been really great talking to you about container security.

[0:45:35.7] MK: Thank you so much.

[END OF INTERVIEW]

[0:45:39.6] JM: If you are building a product for software engineers, or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an e-mail jeff@softwareengineeringdaily.com if you're interested.

With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers. I know that the listeners of Software Engineering Daily are great engineers, because I talk to them all the time. I hear from CTOs, CEOs, Directors of engineering who listen to the show regularly. I also hear about many newer, hungry software engineers who are looking to level up quickly and prove themselves.

To find out more about sponsoring the show, you can send me an e-mail or tell your marketing director to send me an e-mail jeff@softwareengineeringdaily.com. If you're a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level, then look at sponsoring the show through your company.

Send me an e-mail at jeff@softwareengineeringdaily.com. Thank you.

[END]