CONTAINER SECURITY BEST PRACTICES

In this Run and Reinvent podcast, I'm joined by Maya Kaczorowski, a product manager at Google, to discuss container security. Below is a condensed transcript of the discussion.

Ajoy Kumar: I know a lot of folks know what is a container, but how would you describe to someone who is technical but not that familiar with containers?

Maya Kaczorowski: If somebody knows what a container is, you know it's taking your application, packaging it with the libraries and binaries that it has as dependencies together. That makes it easier for you to move it around your infrastructure and deploy it on other pieces of info that you have. You can develop once and deploy multiple times across the infrastructure. Form a security point of view, it's just about having more consistency in that environment.

Your DevOps will like it because they don't have to do, hopefully, as much work to deploy the same thing in many places. But from a security point of view, it means you can pull out the same thing in many places, which means fewer security reviews, less things that you have to worry about patching, having a more homogeneous, consistent environment.

Ajoy: What do you think, containers are more secure or less secure than VMs and some of the common things in your infrastructure? How would you compare container security with VM security, for example?

Maya: I think they have the potential to be more secure. So, there's only a couple of things that really change when it comes to true underlying security between containers and VMs. Containers don't

historically have the strong isolation boundary. So, containers don't actually contain, despite the name. So, they're not meant for untrusted workloads. That being said, there are new projects like gVisor, like Kata Containers, and Nabla Containers, they are specifically meant to provide better isolation for containers. So, that's kind of going away and changing. Some of the other things that are different is just the industry isn't there yet necessarily, so just like how you'd want to have an IDS or an IPS for a networking tool, or an identity management tool or whatever for your VMs, you're going to want to have the same thing for your containers. But those don't all exist yet. There still isn't a ton of choice in the market for users who are looking for those solutions for containers specifically.

Ajoy: What is gVisor?

Maya: gVisor is a sandboxing technology that Google developed and open-sourced. It's basically a way for – it's a sandbox container runtime, to let you run untrusted environments on the same host as a more trusted workload. So, the idea being if you're running a multitenant environment, you don't necessarily want somebody who's running something potentially malicious for that process to escape and affect other things that are happening on the same node. Like I said, traditional containers aren't sandboxes, are not meant to be sandboxes, so gVisor is supposed to help prevent and provide a security of layers of the sandbox behind that. It's effectively emulating a kernel that each individual container would talk to, but in guest mode. So, you actually have some virtualization-based isolation between what's happening in your containers.

Ajoy: What tools should one be thinking when one thinks about container security?

Maya: How we're thinking about container security at Google is trying to think about this comprehensively. So, if you think about the kinds of problems that you're going have in your environment, and how – and what kinds of threats you're going be seeing – for example, one set of threats is going be things that people are trying to do to your APS server, so if you're running an – because orchestrations are from Kubernetes, people trying to get Kubernetes to do something that's bad. This is about protecting your network, your secrets, your identities – all of that kind of stuff. And there, it's not so much about tooling, as much as following best practices, putting in place what Kubernetes suggests having good secure defaults, templating things that make sense from the get-go.

The second area that we think about in terms of container security is what we're calling the software supply chain. So, how you make sure that the images that you deploy, the container images that you deploy, don't have any issues, don't have any known vulnerabilities, meet your own internal requirements for compliance, are signed off on by your testing team, whatever it happens to be. And in that area, you probably need some new technology around how you verify those images, how you store your images, etc., before you deploy them into your environment.

And the last area that we're thinking about is – we're calling kind of runtime security, things that happen once you have your containers up and running into production. So, somebody is trying to dos your containers, flood your event pipeline, or the things that I was talking about is being – people are quite worried about container escape rate. What happens if you have a workload that tries to leave your container and affect somebody else's container. And in that case, there's also – that kind of traditional idea set, the IPS mindset for containers. How do we build the tool and make something there that works in that environment?

Ajoy: How do developers figure out if they have zero-day vulnerabilities in their software, or in some sense, any vulnerabilities?

Maya: Common vulnerabilities would be documented in the CVSS database, it's the common vulnerability database for security. You have to think about vulnerabilities here at so many different levels, right? So, I'm mostly focusing on container images, but if you think about security as the whole stack, you have to worry about any vulnerabilities in your hardware; any vulnerabilities in your underlying kind of OS operated in a boot, etc.; the actual image that you're going deploy on your hosts in the container world; the image that you deploy within the containers; any issues with your virtualization software; any issues with your container runtime, like Docker or something like that; any issues with the container orchestration, like Kubernetes.

You have so many, so many layers of dependencies and different things you need to protect, I mean all of those could have vulnerabilities. So, for a lot of these, the best practice is to just patch, make sure you're on the latest version of the system. In some cases when you're building your own applications and you have your container images, the best practice there is to scan those images for known vulnerabilities. And there is a bunch of tools that let you do that, both open-source tools and vendor tools, to verify does your image contain any known vulnerabilities?

Ajoy: How does operations really take on Kubernetes security, container security? How should they start thinking about it?

Maya: TI think a lot of the time what I see happen is somebody in DevOps, somewhere in your company will say, "Hey, I want to start using Kubernetes," and will start doing it. Or maybe even in some cases some developers somewhere didn't decide, but their boss' boss' boss decided that they were going to start using Kubernetes, and now they have to use Kubernetes. And this trickles down to the security team, and it trickles down to the operations team and all of that. Form a security point of view, like I said, as long as you're aware that it's going on, there's actually quite a few documented best-practice guides in everything that you're in a pretty good spot.

From the Ops point of view, what changes is your development and your deployment velocity. The whole point of moving to containers is that you can more quickly deploy new things to your environment. So, you're dealing with redeploying entire container images rather than deploying code changes to your database – sorry, to your codebase. So, how you manage that kind of code checking process, that code review process, that deployment process, everything, that might be a little bit different than we were doing already today. The benefit that you have as an Ops person is that you can – you now have a single checkpoint, that deployment mechanism. Or you can enforce certain requirements for the code that ends up in your environment.

So, a security checkpoint might want to verify, for example, is, did I build my code? Or you might want to verify, did I scan my code for vulnerabilities? Etc. But you might also have operations checkpoints you want, like was my code properly tested? Have I tested it on this particular bespoke piece of hardware that I am particularly worried about? Is it valid to be deployed in this particular geography? These are some of the things that containers enable you as an ops person to be able to enforce in your environment.