7 Steps to Engineer Security into Ongoing and Future Container Adoption Efforts

Featuring Rich Laughlin and Tom Scanlon as Interviewed by Suzanne Miller

---

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Suzanne Miller: Good afternoon, my name is Suzanne Miller. I am a principal researcher here at the SEI. I am going to be interviewing two of my colleagues, Rich Laughlin and Tom Scanlon, today about a topic that doesn't sound exactly technical, but it's become technical, which is containers, and we are not talking about Tupperware here.

But before we get started with that, I do want to note that as part of practicing social distancing here at Carnegie Mellon University, we are all operating from our own homes. So, if there are any technical glitches, we apologize for that, but we are trying to do our part for COVID-19 social distancing. We hope that you all can appreciate that. So, we hope that you and all of your families are also well.

We are going to go ahead and get started with Richard and Tom. They are both working on security aspects of containers. They published a blog recently, called 7 Quick Steps to Using Containers Securely. So, I wanted to ask both of you before we get started, tell us a little bit about yourselves. Why don't we start with Rich. How did you come to be doing this work with non-Tupperware containers?

Rich Laughlin: I work under the Security Automation Directorate of CERT. I started playing with containers a couple years ago now. At the time, it was entirely as part of just exploring new tech, interesting tech, and then started to become more and more part of my daily routine, my daily job. It's not been quite a year, but maybe about 8 months ago, I started working on some content to talk about container security and collaborated with Tom on some work for a customer, and eventually it became this blog post. It's been a fun experience.

Suzanne: Let's go to Tom.

Tom Scanlon: Thanks, Suz. I am a senior researcher here at the SEI. I have been involved with containers in a couple of facets. The first is we are helping a lot of programs transition from traditional waterfall approaches to software development to Agile and DevSecOps, and application containers are a huge enabler for DevSecOps. [There are] a lot of reasons we will talk about throughout this talk, but they are modular or portable and allow for rapid development. So, working on them tactically, through those efforts. In addition, I work a lot of programs on sort of end-to-end software assurance from acquisition through sustainment, and where can we put specifically security into there. That is really where I first got concerned about containers because in those talks with folks that are doing program management and purchasing and things like that, they sort of sense that containers were a new good thing, which was nice, but they also sort of thought, containers automatically meant you were increasing security. I have had to educate some folks that containers are a new technology. It has pros and cons like any other technology. It certainly brings with it some security advantages, but just because a program or an effort is using containers doesn't mean they are more secure than someone that is not. Folks were starting to use containers as the answer on compliance check sheets and things like that. *Well, we're using containers, so we are secure*, and so that was part of my motivation to get into this.

Suzanne: Rich, how did you and Tom end up working together?

Rich: Tom and I work on the same team. He found out that I was writing some content up for container security and he said, *Hey, I am also working on a paper related to containers. Let's collaborate, and see if your content can help complete this paper*. That is how we sort of started working together on the topic of containers, specifically with security in mind.

Suzanne: For those that don't understand about containers, what is different about doing software using a container versus doing software without using a container? So, whoever wants to take that one.

Tom: I'll start. What is exciting about containers for me, just from a software development perspective, is because the containers include the software dependencies and the resources you need to run on different hardware, the container allows software to be deployed in different hardware environments with the same container. I remember the days when we did desktop applications and we had to run around each desktop and install the software, configure it, and then hope it ran. Containers are the complete opposite end of that spectrum. If you are running an application and container in one environment, and you switch hosts or switch environments, or move from desktop to cloud—provided that the new environment has a container engine running that supports the container you have—you can easily move the containers around, allowing that

portability. That also allows you to leverage different stacks of hardware you might have laying around. So, instead of old hardware becoming obsolete, as long as it has enough resources on it, I can put a container engine on it and still use that old hardware.

Suzanne: One of the things that in DevOps we talk about is environment parity. So, what you are saying is that containers are one of the strategies for achieving environment parity, which is having the same environment available for doing testing, for doing reconfiguration, for deploying to multiple sites. Is that correct?

Tom: It is correct. In essence, it's actually sort of making environment parity not relevant is what it's doing, because you don't exactly have environment parity, it just doesn't matter because the way the containers are defined.

Suzanne: So, the container is the boundary of the environment that you care about?

Tom: Correct.

Suzanne: OK. Alright. What are the security concerns that come along with this? You've got seven steps to getting to ongoing security, so what is it about security that you have to pay attention to? Rich, you want to talk about that for a minute?

Rich: Sure. Like any technology there is some amount of inherent risk, so if you compare containers to virtual machines, virtual machines are some inherent risk that someone might try to break out of a virtual machine. Similarly, there are some inherent risks for containers that someone finds a kernel vulnerability and breaks out of a container. But, I think, if you were to look at the entirety of your risk, most of your risk comes from configuration problems, using containers in the wrong way. The inherent risk there is a problem, but it is really an issue for kernel experts and people who are working on the kernel to try to make sure that can't happen.

For the regular user of containers, the main thing is going to be supply chain. I could say supply chain until I am blue in the face because it's sort of the main thing you get. When you are building containers, you are building them from something and that something is where a lot of your risks come from. The other part of that is like Tom was saying, is that a lot of people will use containers and think, *Alright. OK, I'm good. I can just stop there. I don't have to ever think about security again*. But all of the traditional kind of security concerns, decent logging and auditing, managing resource limits so things don't take over all of the system resources and bring the whole thing down, that sort of thing. All those things are still relevant.

Suzanne: What are the first couple of steps in your seven-step process for getting secure containers?

Tom: They are not really ranked. They are just steps. I am actually going to start with a small one that we mentioned last in our blog, which is persistent logging, because I think that is a gotcha that gets folks that are new with containers. Containers have an ephemeral nature about them. That is one of their advantages. But, if you are doing all your logging within a container, then the container goes away, either because you take it down or it crashes somehow, you have lost all the logging that you've done that you need now to debug or chase down a problem, be it security or just an operational problem. Pushing those logs to a central repository or somewhere out of the container is advantageous to give them some persistence. I really wanted to highlight that one first because that is useful operationally in security.

A second one, which we do mention first in our publication, is just using the resources that are out there. NIST has a nice Application Security Container Guide to get started with. There are some tools that Rich is very familiar with, he has utilized from Docker itself. Docker has some security documentation on their website, and then they have some tools, the OpenSCAP tool, which lets you do some security testing. There are tools and resources out there. So, a good starting point is just get familiar with what's out there. None of these things I mentioned cost money, so you can get in there, play around. There are certainly some really robust commercial tools coming on the market, but I suggest you play around with the open source stuff first, so you know what you are doing when you get into the commercial tools.

Suzanne: Excellent. Rich, a couple of others that are your favorites?

Rich: I think my favorite really is the fact that when you are dealing with containers, you really need a process for rebuilding container images on a regular basis. One of the things that makes containers very different from virtual machines or traditional kinds of deployments, is that you are building an immutable image that serves as the basis for that container. You ship that image around and you run it in various environments. The best practice for dealing with containers is to never do something in a container that you intend to stay around for a long time. Or, you don't want to do persistent stuff in that environment. They are meant to be ephemeral. You bring them up. They run for a while. You shut them down. You replace them.

One of things we talk about in the blog is that what you really need is an automated process for making sure that you are pulling in security updates from upstream. So, you have some base image that you are working with, you need to make sure you are rebuilding from that image on a regular basis because you are not going to get security updates any other way. You could imagine someone might say *Oh, well I'll just have someone, every Friday they will sit down and rebuild the image*, but we get busy, someone gets sick, stuff doesn't happen, and then you end up with security vulnerabilities in production, right? When you are working with containers in production, it really implies a certain level of sophistication. So, you kind of have to have a DevSecOps pipeline to rebuild those images.

Suzanne: What I am hearing is that you want to make that part of your automation processes, not rely on the human in the loop to actually run that security update, but make sure you are redoing the image as part of your normal automation of that pipeline. Is that correct?

Rich: Exactly. I think it also brings some new benefits as well, right? So, one of the downsides to having a mutable environment like a virtual machine in our traditional deployment, is that when you do those system updates, they could break your application. Then you have an outage of some kind, right? Having that kind of DevSecOps pipeline gives you the opportunity to build the thing, do some testing before you deploy it, and then deploy it to get those security updates out there but also having done some testing so that you know that it still works.

Suzanne: What are a couple of other things? We have got four. So, what are the others that people need to pay attention to? Tom, you want to give us another one?

Tom: We talk about configuring the resource limits. So, the way containers work is they will specify what type of resources they need available to them to operate. That can lead folks to maybe err on the high side and allocate more resources than they really need, just so they don't quote-unquote, run out. But, then, you are enabling a denial-of-service attack if an adversary were to get in. So, you really want to set the resource limits appropriately on your containers. That is another just basic consideration.

Suzanne: I hadn't really thought about the fact that if you leave too much space, that you are actually opening yourself up for attack. That is a valuable thing to remind people about. What else have you got, Rich?

Rich: I already touched on it a little bit earlier, supply chain, so securing the image supply chain. So, you are building your container from a base image. You need to know where that image comes from: *Who is building it? Who are they? Why can you trust them to not put malware in there or something like that*?

Docker actually has on the Docker Hub, they have some images that they build and provide and they sign them. You have to do some extra steps on the Docker daemon to actually have it enforce that it is using signed images only. You want to really make sure that you are using images that come from a source you trust. And, ideally, you would be checking the signature on those images to make sure they are coming from actually who you trust and not someone else.

Suzanne: Some of the things you are saying are things that I am accustomed to hearing when we talk about security applications. A couple of them are unique to containers, the idea of being aware of persistent logging versus ephemeral logging, things like that. So, *Here I am. I am new to this*. I heard Tom say, *Get out there on the open source community*, but what are some of the places that you recommend that people learn about this as a way of doing more secure software

building in settings where you need this ability to move from one setting to another, hardware wise, et cetera?

Rich: Some resources that are out there that can help, I think the first one is to pick a container orchestrator. One of the more popular ones is Kubernetes, but there is also Docker Swarm, and set them up in a way that is ideal based on their documentation. Then, leverage some of their features to help you automate away a lot of these problems that you might have. Kubernetes has the ability to set resource limits on resources, so when it spins up pods, it will automatically tear them down if they go over their resource limits. Then, they also cover some of the other availability problems as well. So, the idea that you can have a container run three copies of the same container within Kubernetes or Docker Swarm, and if one of them goes down for whatever reason, it gets whacked because it uses too many resources, there are more copies still running, still available.

Suzanne: Any recommendations specifically from you, Tom?

Tom: I think I will circle back to what I said originally. Just get the tools that are available, play with them, you learn a lot by doing in the space. As you get in there and you start building containers, seeing how their image repositories are set up and how the build environments are set up, you'll become smarter with it, and you'll be able to utilize the security tools better because you'll know what you are looking for.

Suzanne: OK, that's fair enough. What's next for you guys? This is a collaboration that you have started on this one topic. Are you going to continue collaborating? Do you have some other ideas of things that you want to be adding into the work with containers, or are you going to go off in a whole different direction?

Tom: We are probably going to go into some perpendicular directions here. I am interested more in how this fits into the overall software development process. Some of the things I am concerned about are when you get that DevSecOps environment running well, and you are using containers, you are releasing containers a lot more frequently. So, the security scanning tools that you typically use, you are going to have to run them more often because each time there's a new build you are going to have to rescan because the results from the last build are no longer valid. That also means you have to do something with the results of those scans faster because there's going to be a new scan coming sooner. So, what kind of strategies do you have to triage those things and mitigate findings, and to use automation so as containers are coming out, you're checking for security on them, and you doing something about the results of your findings. Those are the areas where I am interested in—sort of how does using containers and using containers securely fit into an overall DevSecOps program.

Suzanne: Excellent. What about you, Rich? Where are you going next?

Rich: I've actually been working on some security automation for Kubernetes for a little while now, it's something that I'm hoping to release soon. I'm going to write some blog posts about it and do some work in that regard to just talk about it. Basically, the premise of that is that upstream, Kubernetes provides some tooling for standing up Kubernetes clusters, but that tooling does not take security as seriously as I would prefer. They err on the side of usability because Kubernetes has a lot of components, and it is a lot to learn when you are first setting it up. So, they try to make it super easy for someone to get started, which has value, but then you decide *OK, I'm done playing with it, now I want to put it in production*, and the tool you are familiar with is not going to give you the level of security that you probably need in production. So, I have been working on a set of tooling to just serve as kind of a reference for, *Here are the things that your tooling should be doing*, and places where people can tweak certain settings if they want to get a balance between usability and security and really understand what exactly they're putting in production.

Suzanne: Lots of Kubernetes users are going to appreciate that.

Rich: I hope so.

Suzanne: I want to thank both of you for joining us remotely today, and I look forward to seeing some of the work that you both have coming up in the future. For those of you that have not read it yet, their blog is available at insights.sei.cmu.edu. The easiest way to find it is to search on one of the authors. Scanlon is easy to spell, so I'd suggest that one. Any resources that were mentioned here, the tools that Rich mentioned, and some of the other things, open source kinds of things that Tom was talking about—we will have inclusions in the podcast transcript so that you will be able to access those.

As always, you can get this podcast all the places that you get podcasts. We welcome you to do that. I want to thank all of our listeners for joining us during this COVID-19 special podcast. I look forward to talking to you all in the future. Thank you for viewing.

Thanks for joining us. This episode is available where you download podcasts, including SoundCloud, Stitcher, TuneIn Radio, Google Podcasts, and Apple Podcasts. It is also available on the SEI website at sei.cmu.edu/podcasts and the SEI's YouTube channel. This copyrighted work is made available through the Software Engineering Institute, a federally funded research and development center sponsored by the U.S. Department of Defense. For more information about the SEI and this work, please visit www.sei.cmu.edu. As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you.