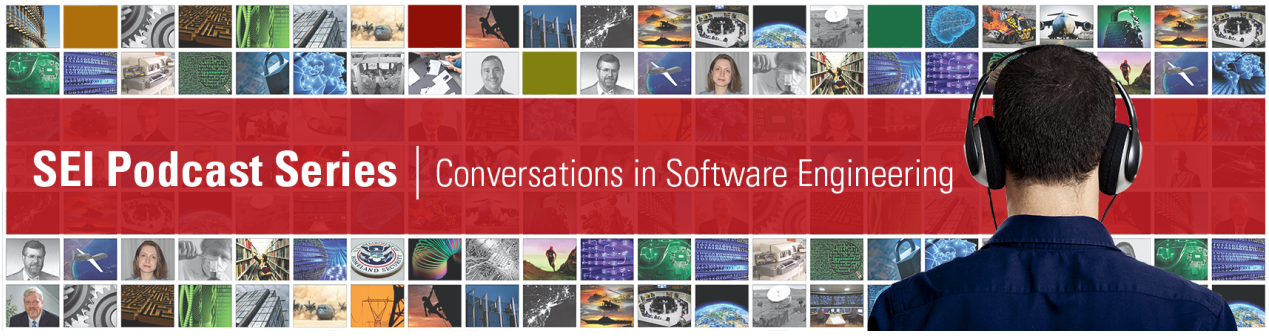

Moving from DevOps to DevSecOps

Featuring Hasan Yasar 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: Hello. I am Suzanne Miller, a principal researcher here at the SEI. This is part of our ongoing podcast series, and this particular edition is coming to you from our homes because we are still in quarantine. So, today I have the honor of interviewing Hasan Yasar, my colleague, my boss's boss, the technical director of the Continuous Deployment of Capabilities Directorate.

We are here today to talk about the move from something called DevOps, which many of you have heard about, to something called DevSecOps, which many of you have also heard about but not everybody has. So, it is time for you to hear about it. I want to introduce Hasan and say welcome. Thank you for joining us today.

Hasan Yasar: Thanks for having me, Suzy. We are part of a team, a DevOps team. Thanks for having me.

Suzanne: Before we get into our main topic, we usually like to get a little bit…even though you have been on the podcast before, if you would give our viewers just a little background on what it is you do here at the SEI and what is it that makes you excited about the work we do here?

Hasan: So, I have been working at the SEI since 2010. I came from industry with lots of experience in software engineering, software deployment delivery. At the SEI when I started, I really bring that experience of the practitioner perspective to the community of DoD.

Now I have been leading a group, such a great group, including you and talking about the Agile and DevOps and DevSecOps and helping the DoD mainly but helping the other industries as well. It is not just DoD but helping the community to improve their software delivery with faster and better quality—not faster and fail, faster and better quality. Now we are calling it DevOps. Maybe in the future, we are going to call it something else, but our intent is always to deliver software faster, better, and high quality. I do like the work because the work is never going to stop. We have been demanding more of the software world. We have been seeing software everywhere. Now, it is time to really look at what we can do to improve the level of the software, the high quality in the various applications. It is not just the web applications, it is everywhere at this moment: mobile platforms, IoT [Internet of Things] devices, and everywhere. We are even talking about the AI [artificial intelligence] component right now, but still, it is software at the end. So, I see it really is encouraging me and also motivating me. The work is going to stay forever, it seems to me, and I am really happy to see challenging environments and various platforms, various components with team cultures and technical stacks. So, I am really excited to continue the work.

Suzanne: I will say, we have talked about this before, every time we solve a problem, there is a new one that comes up that is related somehow to software. So, yes, we are still busy. I have been in this at the SEI for decades, so it is amazing how even though we have solved a lot of problems, there is still a lot to work on. So, DevSecOps is actually I think a nice example of this. For those that do not know what DevOps and DevSecOps are, would you just summarize briefly what is DevOps, what is DevSecOps, and why do we even care that there is a difference?

Hasan: Great question. So, I am going to start about how the DevOps started. So, you came from the Agile community, you know how Agile started. DevOps started in 2009 with the Agile movement actually. It is not against it; it is a side effect of the Agile movement. So, why did that happen? Actually, in 2009, a group of people from the operational team, they were really, not upset, but they would like to do something different in their operational environment to support the needs from Agile teams, which is dev teams that are making more incremental iterative changes. For their applications they are building, their focus is more about customer work. Their focus is more about getting things done, not spending a lot of work on the paper or anything else but just spending more on the content of the work that needs to be delivered.

So, the operational team, they said, *Why don't we change our mindset beyond Agile in operation world, so we can receive the needs from the Agile team?* That started in 2009 as a movement with Patrick Debois and Clay Shafer. So, they got together and said, *Let us change our mindset, make it the developer way to the Agile and operation environment*. They came up with a term as DevOps term in the Twitter handle, and they started DevOps Days back in October 2009. So,

that term actually stayed with the community as DevOps. It was DevOps Days initially. That's how it started.

So, the initial intent was getting the developer and operational team working together, but people realized that, the community realized that it is not enough. Having a more business-oriented, more supporting user needs, required other stakeholder involvement. That work had expanded more including other stakeholders, not the dev and ops, other stakeholders [such as] acquisition people, architects, and the security team, etc.

So, then they built stakeholders, it was still oriented toward business needs and mission. Then it is more focusing on delivering the capabilities. Then we go really fast and fast and fast build, and security people see another side effect of that DevOps movement. Such side effects are not able to test enough for the security needs, not able to test enough for vulnerability analysis, not able to connect the community with the security world. So, security got rather unintentionally forgotten to be included into the DevOps process. You know, by definition of software attributes, we would like to get security as part of architecture, but it did not work out with the DevOps community.

A couple of reasons I can say because when we start to use a more continuous integration concept or a continuous delivery concept, we are mainly using any type of continuation concept, which is very hyped right now. We are using an existing code multiple times in the existing environment like a platform or a backend. The developers start to use a lot of ready code. So, ready code has to be analyzed from the security teams. Developers will likely go or architects they will likely use existing frameworks or using existing libraries and technologies. That type of mindset, again not intentional but unintentionally, opened up the vulnerabilities or mindset into the DevOps pipeline such as, *We are not able to address dependencies*. Such as, *We are not able to address enough security testing*, because we are going to go fast and fast. Because we focus on more customer needs and the customer never says, *I need security*. The customer says, *I need that feature*. That is expected, but as an organization, that is our responsibility to…

Suzanne: They do not say they need security, but if you do not give them security then they are not going to…

Hasan: Exactly. Nobody will be happy to see their data anywhere else. We see a lot of breaches happening almost every day. Nobody is happy to see their information is being sold in a black market. I am not happy, and you are not happy either. Nobody is going to be happy. We are not saying that as a customer we need to protect our data, but that movement was not properly addressing that mission-specific security or specific compliance perspective. So, while we are doing a lot of great work, and people like to see the great impact, we start to emphasize more security into context of the DevSecOps.

So DevSecOps started adding security into every phase of the lifecycle. It is not in one element but adding security into every phase of the lifecycle. That is how DevSecOps term started.

Actually, I have another story on this one, and I have been participating in many DevOps conferences, you know that. Then I have been also organizing and running a couple of events with the community together. So, as part of the big convention for the security convention, which is RSA. The RSA Conference has been happening almost 20 years. So, we had a day for the DevOps Days we ran as part of the RSA Conference, but we were calling different DevOps terms, like secure DevOps, rugged DevOps, we call it many definitions.

About three years ago, as a community, we said, *I know there's a lot of people that are calling different things, rugged DevOps, secure DevOps, DevOpsSec, SecDevOps, something*. As a community, we asked the audience, we said, *What are you going to call it?* I know everybody is calling different things. About 750 people joined our polling and everybody liked DevSecOps. Since that day, we are using the DevSecOps term as defining that concept of adding security into the DevOps process. So, it is not in between dev and ops, but it is basically covering up all lifecycle. We can open up more. That is when the DevSecOps started actually.

Suzanne: I like that as a term because it is really saying security is embedded inside dev and ops, it is not outside of it. I suspect that is one of the reasons that it was popular. What are some of the challenges when you start making this shift, both in terms of organizationally… From your product viewpoint, what are some of the challenges of adopting a DevSecOps mindset, especially if you have already been doing DevOps, what are some of the things you are going to have to think about changing?

Hasan: Honestly, Suzy, based on the survey we have been running almost five years, if an organization has DevOps as the maturity level—when I say the maturity, like, not really implementing one component of DevOps. So, let us open up DevOps a little bit, so I can cover the implementation details. So, when we say *DevOps*, we are looking for the full lifecycle perspective. If an organization has just an integration server, such as Jenkins, with repository management, with the tooling perspective, it is not DevOps. It is the one element of the covering up continuous integration, maybe just not a continuous integration, just integration of the code.

When we are adding a continuous term into that definition, we will likely see a continuity into the lifecycle. *How I set my requirements. What is my case I am working on? How many sprints am I running? Which sprints am I in?* Based on that continuity, we will likely see the continuous integration process for any code changes. Then we will look to deliver the code we built that will go to the staging or any test harnesses. When we are automatically deploying the production that is what we will call continuous delivery.

So, this is all connected with either a tooling perspective or the process perspective. The process is running an Agile process. Process, *How we are going to do iterative and incremental development?* Technology pieces are, *How we are going to architect our component that will support iterative and rapid development?*

So, all this continuity is really helping to eventually add security into it. Based on the survey, we found out that if the organization has good maturity and the DevOps perspective and practices with the process and technology, they were able to add security into their DevOps process more than 300 percent. It is interesting, not a couple of times, 330 percent they were able to get better or faster adding security. Why is that? So, starting from the ground up, adding the security in their components is not going to work out. If you have a good DevOps environment, it is very easy to add a security component into it.

Why is that possible? Multiple reasons we can open up. One is we have a team structure that is already available so we can add security, they use Scrum if they need to. We can add PI planning. We can add sprint planning, post-mortem, etc., the process section of it. We have a platform that the security people can share the information with the team members, easily [they] can share that. So, now we are covering up the technology pieces. Now that we have automation in place, which is the integration or the testing or automation perspective, that is what the security people would like to see as artifacts. So, if an organization has DevOps, they were able to easily add security into it and now we are calling it DevSecOps. We saw that as based on the statistics.

Suzanne: One of the things that in working with my customers I know that the security people have appreciated, is that because we have this idea of a continuous pipeline, we are testing the code many, many times as we make changes. That builds a body of evidence for the security community that vulnerabilities have not actually entered in. So, even though we are changing the code, the fact that we are doing all this testing all the time gives them confidence that many aspects of security testing, not all of them, but many aspects of security testing, they can kind of check off and say, *Yes, I have confidence. I have evidence. I have data to support that this aspect of security has been dealt with in the way that they built this code*.

The security people that I have talked to said that that opens them up to doing the more creative, interesting threats that they do not always have time to think about because they are really just dealing with a lot of the mechanics. That is one of the benefits that I have seen with my customers. What are some of the other things that you have seen in terms of the benefit to security?

Hasan: There are many benefits. Before going into the benefits, I forgot to add to the previous question. There are some roadblocks still, which is the trust problem. Think about the two

communities. Typically, I play the role of a developer, and I kind of play that role as security. I was thinking, I think we should do some play with how the interaction between developers and security.

Suzanne: Role play.

Hasan: Right. I think we should role play sometimes. So, playing that role for myself—and I realize that also teaching the software security class at CMU, I know how the security folks are thinking—there is a trust problem between two entities. Why? Because, first of all, security people are assuming that the developer knows the concept of security. They know what the vulnerabilities are. They know what the best coding practices are. They are expected that they should know all the network stacks and IPs and etc. But in reality, the developer is not trained with security. They are not trained. They do not know. I mean, I have students…

Suzanne: Unless they take a class like yours.

Hasan: Exactly. So, if they have time, probably yes. But in my class, and I have about 30, 35 students each year, I ask that question all the time, like most of them are the CS [computer science] department graduate and good universities all around the world, including CMU. They do not know the concept of security. I know they know at a high level, but they do not know what needs to be done with respect to the technology, with respect to the tools. Also, they do not know what is the specific networking and stuff because they train writing code or building systems with architectures, or good database design, or good algorithm, or anything else.

But security is a living thing, Susie. When I say living thing, it is not as the study guides learning some language, it is a living thing because you have to really follow that up to speed, come up with new technology, a new concept, a new idea. So, the hacker community or adversary communities are very up to date. It is really difficult for a developer following that, what is happening as a trend, understanding all the vulnerabilities, all the possible thinking implemented in the code. So, that is kind of like, without knowing each other, now becoming developers are blaming the security folks as the gatekeeper and blocker. Security people are blaming the dev team for creating crappy code. It is basically, crappy code is bad code. It is creating evil thinking between both parties. But both are intent on delivering good software. The developer's intent is really fixing a lot of functionalities, and the security people's intent is to make resilient systems. So, they do not really converge, and how are we going to build that trust, as you said, in your statement, Susie, we can build up the component so the people can trust each other. So, are they going to trust each other? That's what comes about in the picture because as the security people, if I know developers are working such component, which is libraries and code, then I can see a full transference like an application lifecycle including the artifacts, including the backend stuff, it makes me comfortable what developers are working on. So, I am not going to look at the

developer in a suspicious way anymore because I can see how they are working, what they are working, what type of tools they were using. Maybe I can give them an open kind of environment, they can use the existing library that I have already verified as security people.

Basically, transparency is building up trust between the two entities. The second thing is also with automation concept, automation testing, and analysis stuff, now the security people can see how we are able to speed up the process by creating artifacts as security teams. Also, another thing that is important, the developer will use the existing operating systems or the boxes that the security teams are advising, it will then give them enough comfort to say, *Hey, I am a security person. I have already given them what operating system that they are going to use.* The developer will use that, they are not going to use by themselves and pulling somewhere on the internet, maybe going to Docker Hub, link something from Docker Hub, which is creating kind of trusted sources, and security people will use that. So, the biggest, biggest benefit for me, for both entities, is creating a trust, which is important. Trust is based on transparency and visibility. So, they can see what is going on, they have full visibility of the pipeline, and which is creating a lot of good input for security.

Another important fact, and I am now going back to the other direction, so whenever the security people were analyzing the code, they were able to send the feedback to the dev team in a digestible way. If there is no DevOps, which I worked in for years, when I send my code to the security teams, they were analyzing the code, they were sending me 50 pages of the pen testing results. Do you think that people have enough time to look at the 50 pages of the report, Suzanne?

Suzanne: …Eyes rolling in their heads.

Hasan: Exactly. Because nobody has enough time. With the mindset of DevOps now, we are using a lot of static analysis tools or dynamic analysis, whatever tools we are using, we are able to create feedback on time and in a specific context. Another important factor, as I said this at the beginning, Suzanne, we were expecting that everybody knows everything. It is a wrong assumption. Nobody knows everything. I mean, if everybody knows everything, I am happy to hire. We do not have those people. You should not be that either, because we are not going to create a person who knows everything, it is going to cause a problem eventually. But we would like to increase our team knowledge. Team knowledge should be institutional knowledge in the organization that we are building.

So, with the feedback mechanisms, with the right tools, sending feedback through the chat window, it can be any type of mechanism they are using. The feedback has a couple of components, which is almost all that static analysis has: result of the analysis, what the problem is, what the solution is. So, technically we are teaching the developer what the solution is, so we

are building the knowledge. Now security teams instead of sending 50 pages of documents, now the system is sending their information in a way that the developer will know what the problem is, how to solve the problem, which is another great benefit which is continuous feedback we are calling it. Now, we are expanding feedback equally into security, now the business, now the user, now the security teams are able to send feedback [in a timely way] to developers what needs to be done.

Suzanne: So, these are all things that the automation of the tool ecosystem, I mean, it is an ecosystem now, it is not just a pipeline. But the automation inherent in the tooling ecosystem is one of the things that has really made a lot of what you are talking about possible. When I was writing code in the ’80s, that was not feasible. I remember the very first automated testing software that my company brought in for us to look at. Everybody would just laugh at it if they saw it today, how cumbersome it was, and 50 pages of things to look at. It was not a useful way for us to learn about solutions to different kinds of problems.

Many organizations struggle with this. *What are the right tools?* Do we help with criteria, are there ways that we can help them to not say, *Oh, you should use Jenkins or you should use this,* but help them understand what are the choices that they need to make when they move in this direction? Are there criteria? You can say things like*, your toolset is going to have, low vulnerabilities*. *Okay. That is great. But if I am evaluating an offering for doing containerization or something like that, how do I know? What kind of criteria do I use to say this is going to be more secure than that?* Do we have artifacts, resources that we can help people with in that area?

Hasan: Exactly. So, let us open up that question a little bit further. So, I think, as we said at the beginning, when we get a DevSecOps, we would like to have a software-delivery cycle should be in place, which is DevOps we are calling it. But what are the principles of DevSecOps? The principles are communication and collaboration, the principle of automation, infrastructure-as-code, and monitoring, which is sharing. So, all these concepts [are] the must-have things. If I am picking tools, if a tool is not able to communicate with me in a proper way, which is I am trying to address the communication principles of DevSecOps. So, how can we address the better communication and collaboration of the tool I am using?

Now, we are saying try some principles of DevSecOps. *How can I do automation in the DevSecOps pipeline,* or *how can I add the monitoring concept into the pipeline?* So, all these principles we have to achieve in our DevSecOps pipeline. If I am going to pick the tools, we have to say, *What is the reason why I am going to pick these tools? In this integrated-pipeline concept, what is the puzzle that is going to fit into my pipeline? When it is going to fit into my pipeline, can I integrate that into my pipeline, which is integrability? Can I get data from those tools for a monitoring perspective? Can I use the infrastructure pieces like standing up the tools*

if I need to? Can I deploy independently? Can I maintain that tool independently, which is interoperability?

Suzanne: *…and can I swap it out?*

Hasan: Exactly. Right. *How flexible is this?* So, tools should give you that perspective. So, I pick the reason that I am going to use. Let us say, I am going to pick a tool, the tool will do static analysis for me. Great. So, now I am going to ask a question myself, *Can I add the static analysis tools into my pipeline? Is it going to fit with my build server? Maybe a Jenkins, maybe TeamCity, maybe Bamboo, it does not matter what it is. Can I integrate it into my pipeline? Can I integrate into my ID* [integrated development] *environment?* I saw some tools are capable to really give you feedback early in the lifecycle.

The earliest feedback we can give in a lifecycle is when the developer is writing the code. When the developer is writing code, they were using an IDE, integrated development environment. So, IDE is a good place to integrate the tools as an example. If I am able to, as a kind of tool owner, if I am able to give feedback to the developer when they are writing an object or a code, give the syntax error, give some type of code practice when they write the code, this is the perfect way of shifting left to see the concept all the way to the left side, which is [earlier] in the lifecycle. So, now I am looking for a tool perspective, so, *Can I integrate here? Can I integrate there? Great. I integrated. Am I able to pull data from that tool so I can monitor quickly in my lifecycle*? Now you are addressing a monitoring perspective. *Now, can I share the outcome of the tools into the various collaboration portals like issue-tracking systems? Can a tool create an issue for me instead of creating it manually?*

Now I am connecting the issue-tracking systems. See now we pick the tools, we put them in place, we are asking the principles of DevOps*, How can it increase the communication? How can it do automation? How can it do a monitoring perspective*? If I am able to address all these common principles, that is the biggest criteria. But what we started first was we start with, *Why are we going to use this tool? What is the purpose*? It is very important. We do not want to go to tools that we like. We do not want to go to tools that everybody is talking about in the communities.

Suzanne: Everybody is using Cucumber.

Hasan: But what is the reason?

Suzanne: How they come up with these names, I have no idea.

Hasan: Right. So, like exactly. Like we use the tools for our needs. I have an analogy that I have given many times before, I always describe when I teach the class, I say, when you go to Home

Depot, you are looking for tools, which we do all the time when working our house on some little projects. We always look at the tools that meet our project needs. We are not paying a lot of money. We have a project in our head. We all have put a list of it, we are just buying the tools what we need to for a specific project. It may not be exactly the same thing, but in software, it is the same world. We have a project. We have to look at, *Why I need to do that. What is the purpose? What is the outcome?*

Suzanne: What you bring up there is actually really relevant because there are people—I have some in my family—that go to Home Depot with a project in their head, and they get completely distracted by all the tools that they did not know were available to do something like that. Instead of coming home with a screwdriver, they come home with this automatic thing that dials around, dial in what you want, and the right screwdriver thing pops out. It is like, *I just need a flathead screwdriver. What are we doing here?* We see that in our tooling environments in DevOps and DevSecOps where all of a sudden people have said, *Oh, we have to do that that that that that that,* this whole list of things, and nobody is asking the question that is important: *Why?* So, yes, I definitely resonate with that, and I love the example because I resonate with that too.

Hasan: Today, we are creating a lot of technical debt. Either keeping it in the garage, never using it at all, not effectively. In the software world, in the worst-case scenario, we can sell the tool, in a yard sale probably, but in the software world, it is done. We cannot sell anything. It is done. You are part of it. Then the biggest, biggest challenge actually in the DevSecOps is the sustaining of the environments, Susie. We build up the tools, we build up the pipeline—who is going to maintain the pipeline? Who is going to update that? If we distracted ourselves a lot…

Suzanne: We never talk much about the fact that it is not just the technology environment that changes, but our threat environment changes. Ransomware did not exist 10 years ago. Nobody thought about the way that you hold somebody hostage or hold their data hostage to get money out of them. But now, that is a thing that we all have to deal with. I do not know what the next thing is, and I am not sure I want to. But, there is always going to be something new in the threat horizon and threat environment that we also have to pay attention to. This is one of the reasons this is a never-ending problem set because there are always going to be people thinking of, *How do I get around the blockade that you have put in front of your code? How do I get around that, so that I can do something that you do not want me to do?*

Hasan: Right. It is a good point, Susie, like what is next? When we say, *What is next?* we would like to be resilient enough. We should be ready for any breaches down the road. So, how can we be ready for any type of disaster situation like a breach? Maybe ransomware is one of the examples, maybe other breaches. In the software world, to be ready for any type of breach, we have to know what we have of our assets, which would be our code. If something happens, we should be able to respond quickly for any breaches. The response may be fixing code, the

response may be a patch of the software. If we look at any breaches—I analyze breaches in my study as well, most of the data breaches are based on the common vulnerability, which is a maximum of 24, Susie, maximum 24. All of 24 mainly are input sanitizations, very basic in the CWE [common weakness enumeration] examples. So, that is common thinking. But to fix the very basic vulnerabilities, just the input sanitization, we have to look at the code. We have to recompile, push the new version of it. So, we have to really know that. In the context of the software world, Susie, you have been in the world for almost 30 years, when we switch the context of the different tools or different program or different projects, do you remember anything six months ago? Say no.

Suzanne: Especially in COVID world, I do not even know what day it is sometimes. I just know that six months ago is pre-COVID.

Hasan: Right. For the coder, for the developer…

Suzanne: Not useful, as you say, not useful if I am trying to fix something I did six months ago.

Hasan: Right. So, now we need to have the context. To get the context in the software world, we have to know. We cannot go back to six months. We cannot do time travel, but we have to look at what artifacts are, what tests we have done, what the dependencies are, what the version is, so we can quickly look at our code and fix those specific, very basic vulnerabilities. That is required…

Suzanne: That goes back to what you were saying earlier about we have to have that transparency. If I am a security person who understands input sanitization for this particular type of dataset, I probably understand that better than the actual coders. So, if I can look at what is out there, if I can look at the code, and I can see what is there, I can help them to say, *Oh, this is the spot that probably caused this problem. Let's fix that*. Where it may still take the coder who originally did it a longer time because they do not have the same mental model of what it is they are looking for. So, this teaming of the security and the coding community and facilitating that through these many, many new ways of communicating, I think is one of the things that is making a difference in the environments that I see.

Hasan: Exactly. We did not say another word, it is not a buzzword, it is a real word, it is *traceability*. DevOps is giving traceability. It is exactly we know where we developed the code, which components are, what type of library has been used. We can automatically fix it or pull up and then push the new version. So, we are racing. Another thing I have to say, Susie, that unfortunately, I do not want to say it dark like, the communities, adversary communities are sharing more, but as a team for the software developer engineers or the good people we can say, we are not sharing enough information between us. We do not have that type of sharing

mechanisms, but adversaries are sharing a lot. They are faster, they are sharing. So, we have to be proactive in the concept of security. Proactive means we should be ready. We should be resilient. We should be sharing information amongst each other, so we should be ready if something happens.

Suzanne: Which brings up another trust issue, which I think is one of the things that prevents the people that are trying to protect their environments is, *Who do I trust, who is it safe to share with?* Now you get into the whole bunch of different layers of trust and a whole other topic is the zero-trust architecture kinds of stuff, but I think there are and I think the kinds of things that we do with the DevSecOps Days and things like that, conference events, we know that there are bad actors that go to those just like there are good actors that go to those kinds of things. I think the resilience idea that you are talking about is like, we have to just accept that and understand that whatever we say to anybody can get perverted. But at the same time, we can't be so paranoid that we do not say anything to anybody because then we are going to miss the new ways to protect ourselves in different areas.

I do not want to oversimplify this for people, this is a complex area to work in. But, we are paralyzed if we do not do anything, and we are not able to take advantage of the wonderful technology. Just look at what we are doing right now. Five years ago, if COVID had happened, I cannot even imagine how we would be having this podcast. If there was not a certain amount of trust that we had with whoever provides the video and the audio and the data going across the links, we would never be able to do this. So, you have to make a decision that you are going to act in a way that is resilient but is not a way…You cannot be naive but you cannot just stay paralyzed either. Sorry, I got on a soap box there.

Why don't we wrap this up by helping the people that are going, *OK, so what do I do next?* If I really have not been thinking about DevSecOps, what should I be doing and what kinds of resources should I look to the SEI for to help me in making this transition, because I need to make this transition?

Hasan: We have so many resources either through the SEI or through the communities. So, we have space for DevOps, we have space for Agile. I really encourage all our listeners to take a look at SEI resources. We have been publishing webcasts and webinars, blog posts, and technical write up, technical papers, research papers we have been publishing. Also, as you heard from Susie, we have been hosting DevSecOps Days. Another one is going to come on October 1st as a virtual event. So, please participate at those types of events. As attendees, you will get a lot of lessons learned from others. But we have to be really careful because we do not want to be exactly copying and pasting the same infrastructure—we should remember those *why* questions. *What is my project? Why am I going to achieve that? What is the purpose of it?* We should remember that. Based on what we know, what we have internally, let us build up DevOps. If you

do not have a DevOps, start to build a DevOps environment. When we build up DevOps, include security folks at the beginning into your pipeline. Either a process with the team cultures or the tooling perspective. That has to go together parallel. Do not just only build up the pipeline and say, *I am going to build up the pipeline first and then go look at the process later on*. It is a wrong assumption. Or, just covering up the process without getting into what is wrong. So, we have to go as parallel. We have to go, like, raise up and raise up and we can get better on the three classical terms process and technologies and then talking about the people's actions. So, we have to go together.

We have to train the people. We have to have the right processes in place. We have to have the right technologies—it is a balance, so we can increase the bar. For DevSecOps, there is no limit on DevSecOps. We can get better and better and better. Because it is a journey, we are just looking for what things are not working, we can go fix it. Sometimes we may see some training problems. We may see some tool problems. We may see some process issues. We may see some compliance perspective. So, we may see many types of barriers, and we have been able to see the barriers based on what we see and then that is what enablers are.

Suzanne: All right. So, I want to thank you. I always enjoy having a conversation with you about these topics. I think the next one might be on what are the future expansions of that middle part, because we are hearing about people saying, not just DevSecOps but DevOps as…

Hasan: DevResOps.

Suzanne: So, that can be our next conversation. For our viewers, we will have in our transcript links to all of the kinds of things we were talking about, assets of different types the SEI has provided, community, portals, and things like that. So, you can get that all when you look at the transcript. We will also have this podcast available in all the places that you get all your other podcasts, whether it is Soundcloud or other places. So, please do look for this podcast and share with your friends. Thank you for joining us today.

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.