Actionable Data in the DevSecOps Pipeline

featuring Bill Nichols and Julie Cohen

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: Welcome to the SEI Podcast Series. My name is Suzanne Miller, and I am a principal researcher in the SEI Software Solutions Division. Today I am joined by my colleagues and friends Bill Nichols and Julie Cohen. They are also members of the Software Solutions Division. Today we are here to talk about their recent work looking at actionable data that can be pulled from the DevSecOps pipeline.

Welcome, Julie and Bill.

Bill Nichols: Well thank you.

Julie Cohen: Hi, Sue.

Bill: Glad to be here.

Suzanne: This is going to be a fun topic, but before we get into that topic, let's begin by having you tell us a little bit about yourselves. What brought you to the SEI, and the work you do here, what is cool about it? Julie, you are new to our podcast series, so why don't we have you start?

Julie: My name is Julie Cohen. I have been at the SEI about 20 years now. Before that, I was in the Air Force for 20 years as an engineer and a program manager with a lot of different jobs. When I left the Air Force, I had a couple of jobs in industry. Then I joined the SEI where I have worked space programs and non-space programs, acquisition, engineering. It has been a good ride.

Suzanne: What is the most fun about what you do?

Julie: Actually helping the customer—seeing the customers struggling with something, helping them to see different options, and then watching them pick a good option and move forward.

Suzanne: I could have predicted that answer. I have known you for a while. We both have that same sensibility. Bill, tell our audience about yourself, those that have not run into you before, and same thing, what is cool about what you do here?

Bill: My background is experimental particle physics. That is where I actually got my Ph.D. But I spent the next 15 years or so working at a Naval nuclear lab doing the programming for scientific and engineering design-and-analysis tools. I wrote things that were used to design the next-generation nuclear reactors, built simulators, and so forth. As we went into, not only developing but modernizing that, we got involved in modernizing a large suite of analysis software. That is how I got into software project management. It was also good exposure to some of the needs of the Navy as far as being able to meet certain types of schedule demands. For example, you had to have simulators ready for your crew to train on, and software better be ready for that. You had to have the onboard software ready for the crew when you are ready to launch the boat. The schedules really did matter. Not as much maybe if you are at NASA, and you have to meet a launch window, because it is going to be a couple years before you can go to Mars again. But the schedules were important.

We worked with one of the teams from the Software Engineering Institute on how to do some of this program planning. That is how I really got into it. After a few years, I actually came to the SEI to learn a little more about that and to share what I was learning with other people, and to find out what others in the world were doing, see this from a broader perspective. I have been with the SEI now since about 2006, so it has been essentially half of my working career. What I find most interesting is, yes, I really get excited when I

get a hold of the data and can try to make sense of it, understand what it means, think about how you can improve the process or make predictions. It is like, wow, they actually pay me to solve these puzzles.

Suzanne: Yes, you are a puzzle solver. I think all of us are in different ways.

I want to move us into talking about DevSecOps. Just before we get into your topic, I want to remind our audience members, especially those that are new, that the SEI has lots of resources [Click here for all the resources in the SEI digital library], including podcasts, that will provide you with an overview of DevSecOps. We have things from the Continuous Deployment of Capability directorate, we have things from CERT, we have things from your group, we have lots and lots of things. And so we are not going to go into an overview of what is DevSecOps here. We will have links in our transcript to all kinds of good resources. But I do want to talk about how DevSecOps has changed the game when we are talking about the challenges that programs face and why automation is so much more important today than it might have been 20 years ago or 15 years ago when you were doing some of the analysis that you were doing in the past. So I do not know who wants to take that one first, but…

Bill: Well let me take a couple of things on this one. First of all, DevSecOps is all about continuous integration, continuous deployment, infrastructure as code, automation of tests, and builds, and deployment—everything is automated. If you can automate these things, it means you have to be able to describe your workflows and your process with a fairly high amount of precision, which is what I have been advocating for years. There was always this resistance in the software community to doing that, but there has also been that counter-tendency of people love the tools, they want the tools to help them. Well DevSecOps, the tools are doing a lot of that manual work with pushing things through deployment, managing the system. And what has really changed with this level of automation is the precision with which you can actually describe your workflow and your work process, and that provides a lot of opportunities for measurement.

The challenge is, because things are coming through continuously, you are doing this continuous integration, continuous deployment, you have this constant stream of data. So just keeping up with the data becomes a challenge. Everything is being driven by tools. Well, the tools should be able to help you automate that data collection. That presents another set of problems, but the tools then, which are enabling this fast, rapid development, also provide you an opportunity to measure what is, in effect,

a precisely defined workflow. That is what has really changed the game in the last few years—this realization that we do have a repeatable, prescriptive type of process that we can measure and evaluate.

Suzanne: I know some of the work that you did earlier, that resistance to defining a process in precise ways that can be measured. I know from the work that I did in that timeframe, a lot of that resistance came from, *If I define it then somebody is going to want me to collect measures for it*, *and then the collection of measures in that timeframe was not automated.* I remember us having little screen things in the upper left-hand corner just to record time, stop-and-start timers and things like that, that were actually intrusive to the flow of the work, which is one of the reasons that we would see some of that resistance. The fact that we are, 15, 20 years later, able to make process measurement low drag is something that is not just significant for the developers, because I am going to go to Julie now, the managers have always wanted to be able to have the kind of insight that we now can have because we are able to collect the data systematically. Because when we are collecting it manually, it is not systematic. I will raise my hand as saying, I have tried several times in different settings to collect data manually, and I would not pass my polygraph if I were to assert that I consistently collected my data. We are humans, right?

Julie, what is it that the program managers that you work with in the acquisition settings, what is exciting about being able to get this work for that role in the development process?

Julie: I think one of the things that is probably the most exciting is having more access to real-time data, or at least near-real-time data. For those of you who have dealt with earned value management in the past, you get a report that ended the end of the month, but you do not get that report until the end of the next month, and so by the time you get it, the data is almost 60 days late in some cases—very hard to manage with data that is that late; even a regular cost report can be very late. While you may not get precise cost data from the DevSecOps pipeline, you can certainly see effort and how much effort is being expended and those types of things in near real time. Certainly your schedule information is much more real-time, and you can see what was planned and then what was actually accomplished in much closer to the time when it is being done. So if problems start to pop up, you know much sooner. That was always the premise of earned value anyway, was that once you start to get behind you can never catch up, so the earlier you know that you are behind, the better off you are. Now you can probably find that out even earlier. Same thing with product performance. You can see those

tests, and as far as risk goes, you are lowering risk because you are automating those tests, you are redoing those regression tests with every build, and so that lowers your overall risk as well. Those are some of the things that the program managers really will benefit from.

Suzanne: We have all been in the position of having that data that has a time lag to it, that makes it, I will not say useless, but you certainly can't take the actions that you would want to take. *If I had known this two months ago, I would have made a different decision.* To my mind that is what it comes down to is enabling program managers and acquisition people in different roles to make better decisions. If they have real data that is near real-time that they have faith in, then they can make decisions more confidently, and they are going to make better decisions.

All of this of course depends on tooling. One of the challenges I have run into in areas where we are trying to use DevSecOps in productive ways, both for developers, testers, and managers, is that not all tooling is created equal. Oh by the way, the tooling that the development contractor has, and the tooling that the tests lab has, and the tooling that the government oversight groups have are not always the same and not always compatible. Can we talk a little bit about what are some of the key tools and types of data, and are there any insights that you have had in terms of, if you could just make this piece of the pipeline the same tool, that would make so much difference? I do not care about all these others; we are never going to get everybody on the tool. There is not one tool to rule them all. But if we can at least get this kind of tool compatible, if not the same, then things would be much easier. I do not know, Bill, did you want to take that one? You are smiling.

Bill: Yes, let's look at it from this point of view: the specific tool does not matter so much as long as you have the right interface to get the kind of information from this tool that you really want. So whether you are using the open-source version of Jenkins, or one of the proprietaries, the question is what information you really need to collect?

The biggest rookie mistake in most of these efforts is they start off with, *Hey, what can we measure*? And that is always the wrong thing to do for a variety of reasons, but I think the biggest is you end up measuring a lot of things that you just can't use.

The second problem is not only if you measured a bunch of things that you cannot use, but you have almost certainly missed measuring something that you needed to make the measurements you did take useful. One of the

things that we have been finding is that you can get a lot of great information from these tools, like the Jenkins is going to be able to tell you things about whether the builds were successful. When you push these things into deployment from your GitLab or whatever revision control, you can tell things like how many interfaces were changed, or how many files were touched, [or] what went into a specific build.

But what did they build? You just know that a certain amount of code went into a build, it passed some tests, it went into production. OK, what went into production? How does that advance my project? Unless you are thinking ahead to some of the problems, like how am I going to tie this into the overall workflow with something like, *What is going on with my GitLab or my Jira? What are the tickets? How do the tickets roll up into features? And those features, how do they roll up into capabilities?* If you have not connected those different pieces, you are just getting a bunch of point information that is not really going to be helpful for the programmatics. It reminds me of one of those old jokes: *Here is a partial score, Stanford 24.*

Suzanne: I was reflecting that perspective that I hear a lot: *Can't we just all have one tool?* Or *Can't we have this tool?* But I appreciate your point that in today's world it is about the interfaces. We have made so many strides in essentially externalizing interfaces so that it is feasible for us to have compatible interfaces across lots of different toolsets that that should not…the lack of sameness in the toolchain should not be a barrier to us being able to get the data we need. But it means, as you said, we have to think about what is the data that we need. Now I go to Julie and say, what are some of the types of data that program managers need at various points throughout the software development lifecycle that our audience may not be aware of as being valuable that we can pull from these kinds of DevSecOps environments?

Julie: Probably one of the best things is schedule type of information, where you have planned work, you know what you have planned for that sprint or that increment, and you can now see very clearly what is getting done and what is not getting done. So if work is not getting accomplished, if there is a capability that is having problems and lagging, in the next increment or the next sprint depending on how you are running your program, do you want to prioritize something else and think about that capability a little bit more and what is needed?

Another thing is quality. You can run code-checking tools that can help keep track of quality so that perhaps you are getting less technical debt from the

day that you deploy. There is always some technical debt, but if you are checking your code regularly, then you could be more clear about what the quality in that code is. Same thing with cybersecurity. That is becoming more and more of an issue. These automated tools can help you get a handle on just where you are with cybersecurity and where your risks might be.

The other thing that it helps you look at is stability. Do you have a stable process? Do you have a stable pipeline? Is your flow good? Are you meeting your velocity on a regular basis? Those types of things. Are things working well? Because if you have a pipeline that is fluctuating up and down, and it is always something different, then you probably need to do something. Some of these early indicators, maybe not for the program lead, but for a manager who is involved in the software development, they can get some very good metrics quickly enough to enable them to make some changes if they are needed.

Suzanne: I am going to go on both the points that you both just made. I do want to remind our viewers and especially, Bill Nichols, both of you are involved in this work, we do have a set of techniques called GQIM, goal-question-indicator metric, that are actually designed to help you figure out what are the things that we really want to know about. And we can get into timings, we can get into fidelity. I think that that point of just because you can measure does not mean you should, because even with automation, it costs money to gather data. As you were saying earlier, Bill, it takes time and effort to manage the data, to keep it fresh, to understand what is stale, what is not stale, and what is relevant and what is not relevant. There is a whole system of measurement and analysis that you need to deal with. SEMA is for [software] engineering measurement and analysis, that is the group that Bill works in. There are a whole lot of things that go into this beside just having access to the data. To that point, Bill and then Julie, what are parts of the systems lifecycle where data captured from DevSecOps pipelines, you would say, *You know what, maybe that is useful for some niche over here, but it is not going to be generally useful data, especially to the management.* Are there some types of data like that that you want to highlight to our audience?

Bill: From a program-management standpoint, the kinds of things you are most interested in are things like what are the costs, and what is the actual build up? A manager should not care about the story points. How many story points you are getting, the velocity—totally useless. What you want to know is, how is this burning down my product-increment backlog? What we are finding is even among the organizations that really have their act together, that are running pretty decent pipelines, they will get into the situation and

they will be two months into their three-month product increment, and they still do not know if they are going to get all their features in.

Well that should not happen. But that happens because they have not thought ahead of time what it is they need to measure. They are so focused on individual atomic things like story lead times that they have forgotten how are they actually accomplishing? How are they aggregating these things? How are they aggregating to the actual work?

The other problem is you have to deal with this between teams. A lot of the work, a lot of the measurement that we are doing in the actual world is idiosyncratic to a specific team. You have to have certain ways of normalizing that work. Unless you have put some thought up front into what you think these different components are going to cost, you have no basis for comparisons.

Suzanne: Right.

Bill: That is one of the things that we are definitely finding that there has not been a lot of thought to how you do some of these upfront estimates, how you trace these through the system, how do you update the estimates. Because you are going to get different estimates when you start going through the development pipeline and you get the bottom-up estimates as opposed to management's top-down. You have to be able to track those. And you better not be throwing these just into any old data lake, because I am from the school that says you really have to think about what you are measuring. If you have to solve that puzzle afterward, that is going to be a real heavy lift. And if you want real-time data, well that is not what data lakes are for. Those are for handing it off to some researcher who has grad students and time on his hands. If you want an answer today, you better know what that data is.

Suzanne: Most of the data lakes I run into are really data swamps, because they take you down, they suck you in, and you end up trying to deal with data that you end up saying, *Why did I do that, that is not*…

Bill: Yes, I call that the software-analysis measurement pool.

Suzanne: Yes, yes.

Bill: The swamp.

Suzanne: Oh, very good. Julie, are there any particular types of data that you have run into where you go, *Oh please do not collect that. That is just not going to give you any help at all?*

Julie: Well I think Bill hit the nail on the head with lower-level velocity in day-to-day types of things. One of the things I will say, though, about the research that Bill is doing that is really important is looking at multiple pipelines. If you have a big project, chances are that you have more than one pipeline. The things that are on Pipeline One may need to interface with something that is on Pipeline Two, and there is a date where one set of work is needed to enable the other set of work. Being able to look at information across the pipelines, to know how that integration is going to happen, is really vitally important. We know that integrating software is important to do early and often. DevSecOps in general helps that tremendously. But if you have more than one pipeline, you need to understand what data is in each one and how they are going to come together. How is that work progressing in each of those at a top level is very, very important to the program manager.

Suzanne: I do not know about your experience, but my experience when you have that multiple-pipeline situation is that everyone is…You get this local optimization versus systemic optimization. Everybody is trying to optimize how the workflow and the measurement and everything works in their pipeline, and it is easy for them to forget about who they need to integrate with and look at, *I may need to slow down a little bit, because I am not really going to get a chance to even integrate with this other piece until this other time period. Maybe I need to be working on something else while they are catching up to where we are.*

This is very reminiscent of how we were dealing with manual processes in the '80s of who is catching up to whom, and when are we going to integrate, and where are we going to integrate. But we were doing it from a manual viewpoint, so in some ways, we had more time because it took longer to get to an integration point. We were not as fixed on incremental, iterative, fast iterations of small batches. We were doing some larger components. You had actually a little bit of slack a lot of times to get to that point. What we are seeing here is, we are moving fast enough that those integration disconnects are immediately visible. It behooves us to look at that whole system of pipelines. Is that the direction that your work is going, Bill? Because I have to ask you the question, what are you going to do after you accomplish the things you are working on now?

Bill: Well there certainly is some of that, but let me take what you have been

suggesting and build on it a little. You are probably familiar with the way they recommend doing things in Scaled Agile Framework [SAFe]. They use the release-train metaphor basically. All of these pipelines are going flat out and whatever is ready, you integrate and you put on. That is fine, that is actually the way you should be doing most things as far as you can.

But, in our world, in the DoD, where you have these different contractors, there are going to be dependencies in those integrations. You can't just make those go away by waving your hands. That is where this problem really comes in. That is why this multiple pipeline and the dependencies is especially a problem in the DoD world, where you have real products that are coming together, and you better have all of them or the warfighter is not going to be properly equipped for the adversary. It is very different than you get in a lot of the commercial spaces. I think that is why it has not been addressed previously, because it is just not a problem that is as widespread in the commercial world.

The kinds of things that I am interested in looking at is extending more of this work in digital engineering. The digital-engineering world is really exploding, but they are running into all of these other growing pains. They have all of these issues with the different portions of the work cycle do not work with the next. They do not have the proper interfaces between them. And they have not caught up with, *How does this affect your workflows?* Which reminds me, another thing that we could have touched on in the tooling discussion is, well even if you go back and say we are going to use the same tools, that does not mean you are necessarily going to be using the same workflows.

Suzanne: Right.

Bill: Or using the same kind of accounting systems in the workflows. You have to think through those issues. How are you going to do the accounting? And that comes back again to our GQIM discussion. What are your information needs? It really helps to think ahead of time, *What information am I going to need to make decisions?* That should inform, not only how I instrument my process, but how I do the bookkeeping, the accounting, and the analysis to get the information I need from it. I think that is going to be very interesting for at least the next decade. As we continue to move into the digital-engineering world, more and more of the work is going to be represented in these digital artifacts.

Suzanne: I smiled when you mentioned that, because that is one of the areas that the Agile Transformation Team is very active in right now, in terms of

understanding digital engineering within an Agile development process, and how do we apply Agile within digital engineering to get better results, get small batches out of that. This is definitely an area that is active in measurement. As you know, it is the last thing people want to talk about. They want to talk about how do I get something, and that tends to be…We have to nudge people in a good direction. Julie, what is next for you in relationship to this work? What are the things that you are hoping to achieve as you move forward with this work?

Julie: I am really looking forward to the piloting phase, where we hope to actually use this on some programs and looking at what kind of dashboards are available and showing those to PMs [program managers], both of the programs that we are working with and other programs to say, *What could you do with this data if you had a dashboard like this? and What would you rather see? What data is most useful to you in this type of an environment?* And getting some really good feedback as to how well this can help the program managers.

Suzanne: I take Bill's point about, it is not just having the same tools and even the workflows, but doing your accounting and bookkeeping the same way. I cannot even tell you, Julie, you and I both run into this, where we think we are getting hours, we think we are getting effort hours, and we are really getting calendar hours. And it is like, *Well that does not help me, that does not tell me what it takes to build this. It just tells me how long it is in the cycle*. Something as simple as that can just completely throw off your analysis.

I feel like we are in a *Back to the Future* kind of place. Because certainly 20 years ago there was a lot of discussion, PSSM, the practical [systems and] software measurement, trying to get people to get clarity on defining what your measures are. The SEI 's work in the mid-'90s on trying to get clarity, the things that we did, and all these templates full of, *Here is all the data that you need to understand, is my data the same as your data*? We are coming back around to it. But, whereas before people would look at that stuff and go, *Oh dear, if I do this, then that is just going to be so much work for me to gather and figure out and how am I going to analyze it. My Excel spreadsheet is not going to handle this*. Now we actually have hope of using that clarity to actually get some interesting and relevant information that, as we say in the title of the podcast, is *actionable*.

That is what is exciting to me, is getting to actionable data for lots of different roles. We can have a whole different conversation about how this leads to actionable data for test engineers, for example. But I agree that program

managers have been left out in a lot of ways of the conversation about what to do with information that comes out of the DevSecOps pipeline. I am really glad that you guys are working on this and look forward to some of our other teams working together with you, because we all have different pieces that we know about in relationship to the DevSecOps and digital engineering and all that kind of stuff.

I want to thank you for joining us and talking with us today. This has been a lovely conversation for me. As people can probably tell, I have some passion for these things. I have passions for many things, that is why I love doing the interviews. I do want to remind our audience that we will have lots of links in this transcript to resources that we mentioned and that are related. There is a blog post coming out on this topic, so you will want to look for that. We want to remind the audience that you should be able to find this anywhere that you get your podcasts if you like SoundCloud, Stitcher, Google, and of course my favorite, the SEI's YouTube channel. So please feel free to see us there or wherever you want to get your podcasts. And I want to thank you again 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 do not hesitate to email us at info@sei.cmu.edu. Thank you.