Managing Developer Velocity and System Security with DevSecOps

Featuring Alejandro Gomez 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: Welcome to the SEI podcast series. My name is Suzanne Miller, and I am a principal investigator in the SEI Software Solutions Division. Today, I am very happy to be joined by Alejandro Gomez, a software engineer in the SEI Cert Division. We are here to talk about a recent engagement where he deployed DevSecOps practices to managing the competing forces of developer velocity and cybersecurity enforcement. We'll talk about what both of those are in a few minutes. Welcome, Alejandro.

Alejandro Gomez: Hello. Thank you for having me.

Suzanne Miller: Very glad to see you here. You are a new face for our podcast audience, so let's start by having you tell us about yourself, what brought you here to the SEI, and the work that you do here.

Alejandro: Sure. Thank you. Well, there are many reasons. One that I can think about that stands out is I have been very interested in applying basic research into practice. Usually, you will get one or the other. After being a couple of years in the industry, I have been wanting to take some of the great ideas that have been developed in some of the academic environments like CMU and in the general computing science community and just try to apply them into practice and see what good results can come out of that. The SEI I think has been a great place that has allowed me to do that, both in terms of the people that work in it who are smart and can apply the research, and also know how to code, and also just the clients with which we work with, which have been great.

Suzanne: Yes. You are really talking about the basic mission of the SEI: to transition good practices out to our communities, particularly DoD and related since that is our FFRDC sponsor. So you are in the right place. That is good to have you here. The recent work that you have been doing, let us know a little bit about sort of what is it that led you to need to figure out a way to manage these two forces together of velocity and cybersecurity enforcement.

Alejandro: Sure. I was recently tasked to help a client that we were working with where they were developing a pipeline and had a number of teams working on this pipeline to develop and deploy code to production. There were a number of issues that were going on at the moment, specifically related to security. At the same time, there was a lot of pressure from management to try to get some of this work through and be able to attain some of the goals that they had. This is something that is pretty common in other places, and you see it in the industry all the time. You have these goals that you are trying to set. At the same time, there is a lot of complexity that needs to be managed. As we have seen the past couple of years, security is becoming an ever more important topic that not just developers have to be aware of but also executives and leadership.

In talking about security and the velocity that is being done, I want to go over some of the definitions of these things. For security, you want to refer to some of the NIST secure software development framework that has been developed there. They identify security with protecting the organization, protecting the software, creating well-developed software, and also responding to vulnerabilities. So it is not just trying to lock down everything up, it is also also a lot more dynamic. It is also knowing what needs to be in place first. There are these different components when it comes to security. There are also different layers in which security can be applied to from, as I said, fthe business level down to the software. On the velocity side, this is very interesting because this has been hotly debated for many years. It first started with number of lines of code that a person would code in a day, and that was measured as productivity. Then it developed into, *Well, how many days does it take for you to develop something?* Then we had this concept of points, Fibonacci numbering with Agile. At the place where I was working, we were working with the number of days that it would take to develop something. I would just make a note that there are some drawbacks to this approach— and some of those things were later improved—which is that developers by nature are optimistic creatures. We like to point things to say, *Oh, yeah. Sure. I'll be able to develop that in a day or two*, and it ends up taking four or five days, and it throws schedules off. One definition of velocity that I really like is, was made by DORA, which is a report that is created by Google every year on DevOps. It is an excellent report based on psychological research that they have—not just psychological but also statistical—where they have narrowed it down to frequent commits, frequent deployment, and a fast turnover rate are some of the key things that some of the best companies have that have allowed them to be extremely productive.

Suzanne: I just want to interrupt for a second, Alejandro. When you say turnover rate, you are not talking about employee turnover. You are talking about the refresh of the software?

Alejandro: Yes. So, by turnover rate, it means if there is a bug in production, how long does it take to fix that? Does it take a couple of minutes, an hour, a month? And that is a pretty good predictor of how efficient that organization is.

Suzanne: OK. When you say, velocity, you are looking at these more modern concepts, not just how many lines of code per hour.

Alejandro: Definitely.

Suzanne: When I was growing up, that was the standard. But, even then we didn't like it. One thing you didn't mention is how manually the cybersecurity aspect was being done. When you started this, was it primarily a manual process to sort of check off these elements of the NIST standards? Was it semiautomated, highly automated? Where did you start from that viewpoint?

Alejandro: At the beginning, the organization was pretty typical of what many organizations have, which is to simply silo security as just another department or team that does these things. You will have your development team on one silo and your security team on another silo. Then it just kind of this very factory-oriented way of looking at things where developers will create the code. Then they hand it off to security and so on. This has a number of drawbacks where if you have a tight schedule, and you are developing code, and just before releasing, security looks at it and sees that there is a security problem, then it has to go all the way back to development, redevelop, go through the whole process again. A more modern approach is given by DevOps. Then we can talk about later about some of the added layers that security DevOps provides too. The idea behind DevOps is that we want to, what they say in the industry, *shift left*. So instead of having security at the end of the process, we want to be able to put it more towards the beginning where we can catch errors and bugs and vulnerabilities quicker, and that will also enable a faster delivery. This is not a simple one-off thing. There are many practices. There is a culture that needs to be developed. One of the things that we implemented was having automated tests on the pipeline that will test for security. Instead of having a security team making distributes before a release, we will have different sets of tests, just a battery of tests to run through whenever developers are pushing towards their development branch. We all implemented a standardized template. Instead of having each team just try to implement things on their own, we want to be able to centralize this process. Each team is going to have a standardized set of tests that they import. They don't have to worry about that. Developers can keep coding and worrying about what feature it is that they are doing. They don't have to worry about how these security tests are implemented or configured. They just include it on their build file, and that will go through. That significantly improved the time that it took from development to deployment, and it also enabled security to be able to have these checks to be done more consistently.

Suzanne: One of the things I have seen when people actually implement these kinds of practices is you get two additional benefits. One is you get a faster learning curve because, as the developers get notice of, *Hey, you introduced this vulnerability* early in their development cycle, they can use that knowledge to prevent that vulnerability from other pieces of the system that they build. The earlier we get to them, the more likely it is you are going to have a better developed code set. That is one benefit. The other benefit is, because we do all these frequent commits as part of the development process, and we are running these security tests every time, we are building a body of evidence for the security team as to the robustness of what we are doing. When they do something that is just at the end, there could still be some fragility in the code, even that they missed. But when you have this sort of industrial view of running the security all the time, we get a lot more robust software at the endpoint for them to evaluate. Did you see those kinds of effects in what you did?

Alejandro: Correct. No. You raised a great point, which is that, if there are certain vulnerabilities introduced—by the way, when we're talking about the vulnerabilities, it is not even just what the developer is writing in his or her code. It is what kind of code libraries they are importing or what containers they are using to package their code. Once it gets through into a branch that they are going to release, it can become difficult to say, *OK, just try to trace where that vulnerability came from and then just stop the entire process from continuing into deployment*. So it is definitely a question of how can we make it so that we take it to the source of where the problem is occurring. How can we make it so that the feedback loop, that is where the developer notices that there is something, there is an insecure practice being committed, how they can roll back on that make a quick patch, and push that.

Suzanne: Other things that I have seen, and you talked about some of these but not all of these in your blog post. One of the things you found as a barrier to developers using the security practices in the initial state was that they actually had to log into a different system. They couldn't just stay in their Dev environment to implement the security practices or to test for that implementation. I see that a lot. The more things you make a developer log into, and the more tokens they have on their ID badge chain, the harder it is to get them to actually use those practices and use those tools. But the other thing that I have also seen is that some of these tools generate kind of a lot of what I would call false positives. In terms of, *In some settings, this is actually a defect or vulnerability, but, in other settings, it isn't*. How did you deal with that aspect of it? Because I have seen that be a barrier to developers being willing to use these tools as they go through the pipeline. Can you tell us how you address that?

Alejandro: Yes. That is a great question, and we faced that interesting problem that you mentioned. I would say there were a few things. Number one is just being aware of where the vulnerabilities are coming from. When there are different analyzers that will do the static code analysis for you, and they will raise an issue has a vulnerability. I think the first step is informing yourself of, *OK, I found that this is a critical vulnerability. Who says that is? Is it some standard governing body, or does it come from XYZ project up on a public repo?* Having that knowledge of, first of all, who is classifying this as a vulnerability is important. Secondly, there are a number of ways to deal around that because, yes, there can be false positives. One that I think is frequently practiced—I have seen this in different teams in the industry—is to have a security champion on your team. This is a very pragmatic approach to this where it is sitting back and saying, *We won't be able to educate every single developer in our business as, you know, someone who knows everything about security. Instead, we will just have one person in a six-person team who will go through some training, who will develop a little bit more expertise on security*. This person will have some of that motivation to try to look out for some of these things so that, when developers are making a merge request and are going through the peer review process of their code, a security champion can see some of these flagged vulnerabilities. They will have some more of that expertise to be able to look at it and say, *Yes, this is definitely vulnerability that you have. You have a potential SQL injection in here*.

Another way to look at this is just to—this is a little bit more of kind of the centralized approach—which is you will have a security team, and they review some of these vulnerabilities individually. One of the advantages of this is that, yes, you will have a team of experts who will be able to tell exactly what it is. They can have a database of, *These are the vulnerabilities that have been flagged, which are false positives versus false negatives*. The trade-off of this is that then you will have a single point of failure where now, if you suddenly have a hundred developers and they are creating merge requests, and you are just sending all of these things to your security team, they won't be able to handle all of that plus all of the other things that they need to do. So I definitely think there is an advantage when facing some of these problems., but also a little bit more generally, trying to decentralize the solution, and trying to take it away from these single points of failures, and empowering your team so they will be able to resolve those kinds of problems.

Suzanne: I have seen the security champion work really well on some teams. That is one of my favorite strategies for that. But before you even got to that, you needed to actually get these folks to actually build the automated pipeline that would allow those tests to be implemented at appropriate points. Why don't you tell us a couple of things that you did in doing that because you did kind of a staged layered approach to where you injected security tests. Tell us a little bit about that and why you didn't just throw all the security at it all at once.

Alejandro: Sure. It's a temptation to just go ahead and do that and, *I am just going to put all the tests I can imagine and just make it run through there and say*, *Yes. There it goes. It's secure*. When we first came into this project, we had to look at, *What are the requirements? Are we working in a constrained environment. Are we working in an environment where it is more open?* Depending on the policies and the kind of environment that your organization is facing, that will determine a lot of the tests and the kind of security that that is what you are looking for. There are a set of common tests that can be done like static application testing and code quality. Then there are other things like, *We are using containers to package our applications*. When you are writing your Docker files, you are going to be importing code images from different repositories on the internet. *Are those safe? Do they have some sort of vulnerability*? It is important to be able to catch those. Also, as the Solar Winds hack has shown and a few recent cybersecurity events have also demonstrated, being able to secure your software supply chain is very important. So just knowing what code libraries you are importing, that will also determine those kinds of tests. The idea behind each one of these analyzers and tests is that you are going to be mapping these to your policy. You start from your policy, and you look for the appropriate tool that is relevant to that. Then in terms of implementing it to your pipeline, there are a number of ways that you can do it. I believe that an efficient and flexible way of doing this is to separate out your tests and your policies to become individual stages of your pipeline. If we remember back to Unix how it implements, how it uses pipes, so how you can use the output of one program to be the input of another. That way you can connect these programs together to create a new source of functionality. It is an amazingly powerful thing because you started from the scrimmages. Well, I wanted to take the same principle into our pipeline where we have our individual stages, and we want to be able to connect them to one another. That could even lead to other kinds of functionality, which we can get to later. So this is the principle, you could say, of modularity, keeping things constrained, so that they can be used flexibly. And also, again, going back to the policy and asking, *What is it that we need*, and then finding the right solutions for that instead of just going to that shiny new tool that everybody's using at the moment.

Suzanne: So modularity but also interface management, right? Because the only way that piping really works is if I have publicly understood interfaces. So I can swap out the modules if I get new shiny tools coming in or new code, new functionality.

Alejandro: Definitely. You need to have that kind of standardized API between these modules that are communicating. Otherwise, it just becomes very difficult. Once we make interfaces public and transparent, then somebody else could pick up one of these interfaces and be able to develop their own module. How exactly that module works, not everybody has to understand that. They just need to understand the interface of that module. That will for a very flexible pipeline that different capabilities can be made from it.

Suzanne: One of the things that I am seeing is that need for, the security posture evolves. We see threats evolve. We see a lot of elements of the environment change. I don't want to have to go back and change all of the implementation of everything. I want those interfaces to allow me to swap things out.

Alejandro: Right.

Suzanne: *I have a new static analyzer that deals with this particular special aspect of code that is very unique to some setting*. I want to be able to swap that in. That is one of the things that this approach is going to enable in our continuous pipeline development. You have really kind of been able to…In your blog post, you talk about the advantages that you were able to get in terms of speed, so that is the balancing. We haven't really talked much about the balancing velocity piece. Tell us a little bit about sort of what the business aspects were of this in terms of the results that you were able to get so that people understand what impact this really had.

Alejandro: The business had a very specific mission to try to put vulnerabilities to a minimum. This took a higher precedence than other things like speed. So different organizations, depending on their industry, will have a different priority of their product. You will see very frequently in tech that just the speed of getting to market is the most important thing, and if something is not right, then they will roll back or create something different for it. In other places like government or sometimes in finance or just different kinds of industries that are more constrained, you want to ensure that there is a security and correctness of the product in place without, of course, affecting too much velocity. So, the outcome of being able to implement this flexible pipeline stages, shifting left with the security practices, making security something that is more present in teams rather than just keeping it siloed is that you are going to be increasing the efficiency of your teams. Once you do that, you will be able to also increase the amount of development, which, as I already mentioned, one can quantify with the number of commits and the number of deploys that you are doing.

Since all of these things were occurring through a managed battery of testing, then all of these things would have been done under the auspices of the security team being not content but all of these things would qualify the criteria that are needed to make sure that, *Yes, this product is safe. This product checks all the things to make sure that this is the kind of security that we need*. So That is kind of how it was resolved, and that was the output of it.

Suzanne: What I got from the blog post is you were actually able to deploy into production when, in the past, there have been barriers to that because of the security shortfalls. Is that correct?

Alejandro: Yes, so, the advantages really were at the edge of it. Like I was saying previously, this is a kind of constrained environment. So deploys are going to be done at a very regular cadence. That part of trying to measure how quick your deployments are, maybe it won't be as relevant since there is a certain cycle of deployments that are required. But, in the weeks that are leading up to that deployment, you are going to have much less vulnerabilities, much less kinds of last-minute things that you need to be changing. The result of that is less weekends where developers are working overtime, less stressed out project managers, and much happier product owners with the result that the product is advancing according to the schedule.

Suzanne: I talk about it being fewer vacations canceled.

Alejandro: Oh, yes, and fewer vacations made after each release too.

Suzanne: Yes. There you go. There you go. When we talk about work at the SEI, one of the things we always talk about, this is something you mentioned at the very beginning, is how do we transition this work into other settings. You have worked in a couple of very constrained settings. I am sure there are some settings that you would like to try this this in that have different constraints than the ones that you started with. What are some of the things you are doing to enable transition of these practices? What are you looking for in terms of collaborators to help you extend the use of these practices into other settings?

Alejandro: Sure. It is kind of bridging these two worlds, right? There have been many attempts to try to put this into a set of guidelines of what security looks like. I think a great example that has been made recently, that has made a lot of advancements from previous guidelines was the Secure Software Development Framework as prescribed by NIST, the National Institute of Standards, in the sense that it is broad enough to encompass different kinds of implementations but specific enough to be able to give some hints to senior engineers and architects as to where it is that they can secure their organization and their processes. On the industry side, there is a lot of experience, and there is a lot of knowledge of tools, but maybe not as much about this kind of wider processes that could be implemented. So I am very interested to be able to test more these guidelines that have been made with the actual practices. That involves playing around, you could say, with some of the parameters of it. As I mentioned earlier, this is a constrained environment where security and correctness is more important. What would it look like if we are trying to implement this in a place where velocity is more important? Or, what happens if there is a place where systems can go in and out all the time; and what would security look like in that kind of environment? I am very interested in that. If there are collaborators who would be interested in looking at how we can connect industry with academia and trying to create more powerful, more resilient, more long-lasting software, then I would be interested to talk to them.

Suzanne: So any particular projects that you are working on next, obviously extending this into other areas, other guidelines, but are there other projects that you want to highlight that you think are interesting that we're going to want to bring you back for another podcast on?

Alejandro: Sure. So, in addition to my working here, the great thing about the SEI is that they will always keep you busy and not just on the same thing but in many different things. At the same time, I have been doing some work, doing software reliability engineering for our own data center. It is a wonderful opportunity since not all organizations have their own on-premise cloud, and we have been doing some very exciting things in there. We are testing out best practices in the industry, and we are able to do it in our own environment. That is a great lab that you could work with.

Another one is we are creating simulations with networks and things like 5g or with cell phones, and being able to do that with software makes that kind of work a lot quicker, and you are able to find out new things, rather than if you are trying to do it with very specific hardware.

Suzanne: Right.

Alejandro: So that has also been an exciting piece of work that I am doing.

Suzanne: You are kind of doing software defined networks in a way.

Alejandro: That's right, yes.

Suzanne: Perfect.

Alejandro: Not just in the traditional sense but also in a containerized environment.

Suzanne: Nice. OK. So I see some future podcasts here. That is very exciting. Alejandro, thank you very much for joining us today and for sharing this information with our audience. I know that there are people out there that are going, *OK, I think I may be able to do this, but I might need some help*. So you may get a few calls out of this. I do want to tell our audience that we will include links in the transcripts to resources like the [National Institute of Standards and Technology] NIST guidelines, the secure coding framework, things like that, that we talked about. Those will all be in our transcripts rather than us trying to read the URLs off to during the podcast.

We also want to mention that this podcast is based in great part on the recent blog post. There are more details that we didn't cover on the podcast that you will find in the blog post about how this actually happened. I want to remind our audience that our podcasts are available on SoundCloud, Stitcher, Apple podcasts, Google podcasts, and, of course, my favorite, the SEI YouTube channel. So if you like what you see in here today, feel free to give us a thumbs up and look forward to more from Alejandro in the future, as well as other technologists that we will be interviewing in this series. I want to thank you all for joining us, and we are now signing off.