Integrating Security in DevOps

featuring Hasan Yasar as Interviewed by Eileen Wrubel

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

My name is Eileen Wrubel, and I am a senior member of the technical staff here at the SEI. With me today is Hasan Yasar, who is the technical manager for our DevOps team. Hassan and his team regularly publish on the SEI's DevOps blog, but this is our first opportunity to have a podcast with him. Today, we are here to talk with Hasan about integrating security practices on the DevOps platform. Welcome, Hasan.

Hasan Yasar: Hi, Eileen. Thanks for having me. I am glad to be part of the podcast series. I have done the webinar and other things, but this is going to be exciting for me to be part of this engagement.

Eileen: Well, great. I think we're going to have a good conversation today.

Hasan: I like it. That's my hobby and things, and I would like to talk about it.

Eileen: Great. Before we start talking about the report that we're going to focus on today, can you tell us a little bit about your background and your role here at the SEI?

Hasan: Yes. So, I have more than 20 years' experience in software development. I graduated from EE (Electrical and Electronics) with an engineering background, but I switched to software engineering after that.

Since then, I have been writing a lot of applications and using different technologies. And I joined the SEI seven years ago.

When I joined the SEI, that is when the DevOps journey started for me. Because I brought mainly industrial experience to the SEI while managing many different projects. At the SEI, we have different projects, from small-size to big-size, from one month's work to a couple years of work, different type of work. That ended up , basically doing a lot of industrial practices on different phases.

I am managing the technical software development teams. We do engineering work. We write code for our sponsors, and we are developing prototype profile concept applications. We are practicing DevOps with hands-on experience, not as reading but actually practicing how we do it day to day(?), which is creating a lot of outcomes and a lot of artifacts for building a DevOps portfolio.

Eileen: Great. Great. Why don't we talk a little bit about—just to give some of our newer audience members some understanding of where we're going—can you give us a brief introduction into what DevOps is, and why it's so hard to incorporate security into all of that?

Hasan: Right. That's a lot questions. DevOps is changing from person to person, organization to organization. Everybody has their definitions. So, I'm going to say my definition. It's not based on the Wikipedia word, I think this is specific to experience. So, DevOps is practices and best practices, and also a set of principles, and it goes together with developers and also operational teams. It's not only for developers and operational teams, it includes other stakeholders.

In industry, it's about more than just the dev and ops teams, like a dev and ops team together. It started, initially, as more about the operational team. So, the idea was to get the operational team and work with the dev team, or talking to each other, mainly for deploying a release or addressing other problems.

Our definition is more broad. In the SEI, it [the definition] is based on our experiences. So, we have to include all other stakeholders--whoever is responsible for application development--that aligns with the business goals that aligns with the business objectives. They all have to be part of the DevOps team. That means that we should have engagement with other stakeholders, together working for the same business goals, achieving the high reliability and secure application. That's our definition of DevOps. It may be loaded, but that's based on our experience.

So, how can we get the security into DevOps? There is a lot of misunderstanding in community as wellbecause DevOps is more about a faster and quicker release, based on the business needs, right? Organizational policies say that *We have to get this feature to our customers yesterday. Not a couple days later, yesterday*, which is forcing a lot of pressure for development and operational teams. Developing applications so quick and so fast, pushing the production to user based features because of the market demands. So, in that context--very fast and very quick--

security folks, they don't like that type of change because every change is creating a lot of questions and lot of problemsfor the security operational folks, specifically because security operational folks are looking for the best practices, or they're looking for the controls based on the organizational security policies. That depends on the organization. If the organization is in the financial industry, if the organization is more about the restricted environment, maybe looking for other compliances, they may have different policies and procedures. So, it is very difficult to get the pace of the quicker release and faster release into the lifecycle or the production environment, because security folks would like the test for every change. But having a constant change, there's no way to get security as part of it. It's the number one problem.

Like the Amazon case, there is sometimes a millisecond interval for a quick release. So, security folks are getting freaked out because they don't want to go and change and look for everything from the beginning. So, that is misunderstanding difficulties.

So why It is difficult to do DevOps because of that reason. However, actually, DevOps is enabling getting security into the lifecycle because of the integrated platform, because of other components. It looks like it's difficult, but actually it's easy to get security in DevOps.

How is that possible so we can deep dive and more but principally, there are three components to integrating security?

One is organizational policies and procedural things. It is very difficult to get it because organizations are siloed, and security teams are, , in a different building, and a different team, or totally that they are not part of developers [team] or they are not part of the operational team as well. Or, they are not part of the requirement-gathering team, which is a cultural program and, more about the silo problem.

The other problem is more technology related, because security operational folks are using different tools that are not compatible with the developers. Like, to give a specific example, maybe security operations is using management tools and keeping track of vulnerabilitiesor looking for security controls or risk management framework, *How do they do that in their environment*? That may not be compatible with what the developers are using. Like, if developers are using the Linux-based environment. If the security operational team is using a Windows-based environment, if they're not talking to each other, even though they would like to talk, they cannot talk at the tool level, or they cannot talk about how the developer feels [and be in] a comfort zone in that level. If they cannot talk, they cannot share experiences.

How are they going to share their experiences or their requirements? Most of the time, security folks are sharing documentation via email: *Here are our top 10 controls. Look for the OWASP 10*, or *Look for the NIST framework*. Or, *Here is our organizational security policies*. So, when

we give a bunch of documents to developers, and developers do not have enough time to digest the documents at the level of writing the code, because the developers think *It's not my job. It's the security operational team's job*, which is creating other difficulties.

In summarization, *people are the problem* means the cultural things. *An organizational problem*, an organization is creating more silo-built teams. *Technology problems.* Those are the three main problems that we are seeing.

Eileen: I have the conversation a lot about engaging security and accreditation authorities with software teams and shifting the focus from really being a place where things stop to get checked and bringing that group in at the very bringing, as much as possible, so that they can inform the requirements in a steady way rather than dumping a bunch of controls on a development team.

Hasan: We would like to get security as early as possible into the lifecycle. But there is a misunderstanding, because in a lot of organizations they are thinking, *I have 100 developers*. I *have 5 security operational people or security experts. How can we bring the five security experts and the 100 developers [together]?* or W*ith more developers, we cannot get in sprint planning meetings. We cannot get in daily Scrum*. These are the challenges that we are seeing.

By using DevOps methodologies, which we wrote in the paper, we experimented as well. If we have a fully DevOps platform integrated, that means from beginning to end. Integrated means it's visible and also transparent to all the stakeholders. We are expecting the security folks should be part of the early cycle, day number one. If the application is going to be developed, or new features will be added into the legacy system, or the private systems, as soon as the order comes from the business analyst or the business developer--whoever the decision maker is--as soon as random things come into live— before developer touches the keyboard—the security team can either share those possibilities or the procedural things, or they can be part of the first engagements. Like, in terms of *What are the security requirements this application will have?* Or, *What are the risks that organizations should be able to handle? Like, what are the compliances we should be complying based on application?* That has to be shared at the beginning.

When it is shared, in a level of the developer comfort zone, like not giving the work documents or not giving a huge number of pages. We would like to see how the security operational teams are using issue tracking system or wiki pages. That can be part of the development team as well. So, developers can use a similar wiki environment most likely like everybody else, or they can share information from the security operational team in a text format that can be part of the wiki pages that the developers are using, or use the issue tracking systems part of it. So, we have that one step right there.

Then, as soon as the architect comes in the picture, or the designer comes in the picture designing the application, they can create security requirements for that feature. Again, it's going to be part of that issue tracking system, which is part of all integrated platforms. As soon as when the ticket is created, the developer develops that functions. Then security components will be part of that function as well. So, the security team can see, *OK that code has been developed, based on the security policy that we already shared*.

Now, code goes into the repositories. There can be another component, another check that can be done, over which can be a code review and can be other components, which depends on the organization. like static analysis can be done over there. It is kind of like a code-review process. Again that result can be seen from the security operational team, they can say, *Yes. We saw the requirements and we saw it has been checked already*. So this is the second time we checked.

Now, when a bunch of other developers--maybe six or seven people--are working these different functionalities, when they merge together, more testing can be done as part of the CI server, which is continuous integration server. In the beginning, whoever collects the requirements may write typical abuse cases. They may write typical security testing cases.

Typical security testing is done at the end. However, we can get the security testing, how the security operational teams are testing applications. So, we can move the security testing to the beginning. So, the beginning is like scripting, basically. Security operational teams should share this info[testing scripts], which can be part of the CI server, which is continuous integration server. So we can start to check in a write up that we had before in the script against that application we will develop.

Once we go into the staging environment, we can do more testing. The first testing can be fuzz testing or PEN [penetration] testing, specific to the security operation of data, and goes to the staging and then the final production environment. There are many steps in the lifecycle that can be checked. But security operational folks, as I said at the beginning, they do more at the end, which is too late, because then it is costing so much time in terms of fixing any known vulnerabilities or fixing anything that has been discovered late, because it's going to go back to the sprint plan, depending on what type of application development methodologies they were using.

If it was agile, maybe a three-week sprint. So, it's going to delay three weeks. It depends on time as well, and it depends on the complexity of the component, if it takes more time to code it. So, one single bug that has been discovered at the end is going to cause probably two or three months [delay]. So, that's going to become a zero-day vulnerability for the criminals. They will see it, because it hasn't been discovered yet, *How can we fix quickly. It's going to take so much time.* That's another component of getting a quick and early fix.

Eileen: Right because we know that the further away we get in a lifecycle from the point at which a defect is injected, it costs us more in terms of everything to fix it. So, yes, if we are doing something with real-time security implications, if a bad actor knows it's going to take us three months to fix something, there's a lot of damage they can do in three months.

Hasan: Right, if the damage is done, there are a couple of things the operational team does. They're trying to get more of the security or network boundaries and applications to protect it. But it's not really fixing the application. It's kind of like a Band-Aid solution. *We're just going to Band-Aid here, Band-Aid there,* but it's not really fixing the overall application. Because we are really doing it at the moment, once we discover it. If we design [application securly[at the beginning, of overall pipeline or procedural things, which is the DevOps way, should be open for any type of security vulnerabilities, because we cannot develop applications on 100% secure. We are making it difficult. We are making it very challenging for the criminals. They cannot hack it, or they cannot change it, more(?) control. But our system should be able to respond quickly as a kind of Band-aid. So, if that means we should have traceability throughout the lifecycle, then if the incident happens, it may happen. If you could discover any vulnerabilities at the end, and we have to fix it, and we find a problem.

We should be able to roll back quickly, which is the DevOps way. We have to roll back quickly to the previous version. Then we have to go back and change and look at who has been designing it and who has been touching it, which is complete integrity because we know who the developers are. Because it's kind of like a quick incremental release. We can go back quickly and fix it and release a short time frame. Because we can automate all the deployment procedural things, we can automate the testing. As soon as the code has been changed, that code can be integrated automatically, then go to the staging environment, which is automated. That is basically creating automation and the lifecycle, which makes it much easier to fix quickly. Then again, it depends the policy, things which we discuss, about the DevOps procedural things and policies as well.

Eileen: I really liked what you talked about in the paper. You said there is so much of software security that is about, *We know these types of activities. We know that the threat surfaces look like this, so we built some barriers. But the idea that we build code that bends but doesn't break, even when it's based with threats that we don't anticipate.*

Hasan: Specifically, when I say *bend*, I am looking for *more easy to respond*. We will get the requirements. Typically the risk management team or threat analysis team would then define what type of threats we are getting for our application or organization. By knowing that threat, and by knowing the principles, we can start to develop applications that are going to be available to receive but at the same time we should be able to go back and fix it if we see something else right away. There is a balance. *How much can we write secure code, it's costing so much money,*

probably. Because we have to find out, *What is the right balance? We cannot get 100 percent security at the beginning because it will take so much time.*

It's that DevOps thinking, which means you have to go and look for more business goals, right? If we look for more business goals, then our application should be compatible [with them] to release quickly. If we are delaying the new features, we are going to lose the market. So, how much can we have security. That means we have to make it a level that everybody will agree on. Then, we can pass by knowing the high priority or fixed high priorities, right? If you see a low-priority risk, it is OK because we can go back and fix it easily. That's kind of like a *bends* strategies. We can go back and fix quickly because we know exactly where we can go and fix it. We can try another mitigation that will probably fix the code or make another mitigation to secure environments. So, these are things we can play around within the lifecycle.

Eileen: I read in your piece that Amazon averages something like one deployment per second over the course of the year. Thinking about problems at that scale, the idea of integrating security into the DevOps process and the software lifecycle at the very beginning, it seems like that's the only way when we're dealing with problems like that. That's the only way that we can go ahead and have that kind of reliability and responsiveness to customer needs or to the business goals.

Hasan: My definition, going back to the work, we are building the blocks. In a security mindset in the old days, we have to check everything from the beginning. We have to look at every code. It will take time.

However, if you change the features, and we know it's a small feature, which is how Amazon does it and how other big organizations do it, how they scale it up. We can look at security complications of that feature, which is limited. Sometimes changing the label, maybe adding text to the web pages, or changing small features and adding an architecture component as well. So, we have to check for the security on that feature only. We don't need to look for the overall application from day number one. It takes maybe eight months, maybe years. That is the barrier so far. What we are saying is, we have to check security every time. It's kind of like a continuous security.

When we have new features, we have to look at security possibilities of those features. Then, throughout the lifecycle, there are multiple points for the specific features. So, once we automate it, once we integrate it, we can look for the security from multiple perspectives and multiple points. Then we can achieve that quicker-and-faster release versus scanning the code. It's impossible. We have to look at incremental versions of that code, which is going through the repositories. We know exactly what has been done for next day or yesterday. We know exactly what the code were checked in. So, it's kind of an incremental checking, not overall checking, that we like to do.

Eileen: OK, so if I'm at a software company, and I'm looking to make a change, I'm looking to start integrating security more to the left and embrace a DevOps mindset, what can I do now? What are my next steps?

Hasan: To get started immediately may be very difficult, may be scary. First of all, we have to change our mindset. Because we cannot say, *This is not my job as a developer. It's only for security operational [people]*. No it's everybody's job. From a C-level person all the way to the developers, it's everybody's job, basically. There's a number of things. We have to change our mindset. We have to say, *This is our security. If something happens in the organization, everybody will pay the penalties, not only me. It's everybody*. So, we have to change the culture thinking, the mindset. It may be very difficult, but that should start from upper management, and also from bottom up and top to bottom, so we can find a common ground in between.

I would also like to suggest other things also: changing the culture of this organization mindset. And the culture is the second one. So, sharing the culture. Like security operational folks, they should share what are the known threats for developers. So, they should educate the developers, meaning, *Here is a problem that I found* instead of saying *Your code is not working. Your code is bad*, but we would like to get more feedback from the security team. *Your code is more vulnerable because of this. This is the problem I discovered. Here's how you fix it. Here's information can go find on fixing it*. Because we are increasing the knowledge in the developer as well. You cannot get everybody trained in writing secure code, it's an incremental process. When they make mistakes, they will learn it. So how we let them learn, they have to see the result from operational team. They have to know how they can fix it. They know, right, what are the resources available. Once they fix, they will share that information through Wiki pages probably, or they can share the information and respond in the issue tracking system what they have.

When it is fixed, and how it is fixed, the next developer will learn, *Hey, here is how I fix it*. They can learn gradually in writing a better code, and writing a much more secure code. So, which is learning as well, the component. Other things we would like to suggest are more about the organizational toolset. So, try to get more integrated tools. Even though people are talking to each other, when they go to their cubicles in the office, we don't want to let them share different toolsets, which is creating another silo and other barriers. We are breaking the silos by bringing the team together in the room. When they go to their offices, if they're using different toolset, still they are forcing some silos, we don't want to get those silos either.

Eileen: Right, it creates a discontinuity if the tools can't talk to each other.

Hasan: Right, it's the human, they're going to use the tools. So, it's all connected together, having the tools maybe cannot solve the problem. We are using the tools, and we are putting the

process [in place]. And we are responsible for the cost. So, it's kind of like a whole lifecycle of things, including humans, including processes, including tool stack.

Eileen: A change in philosophy, a change in culture, and some upfront investment with an emphasis on automation and collaboration.

Hasan: That maybe will scare the financial folks if you say, *It is requiring more upfront investment* because they say, *We don't want to spend so much money*. But, honestly, if they find the defect at the end [of the lifecycle], it is costing much more money to fix it [than if found at the beginning]. Also, if the company data is compromised, it's creating more reputation problems or other types of issues, which is creating more damage to the organization versus spending a little bit of up-front money and putting a process [in place] and putting all the controls in the system. Let them work together. Spending a little bit in upfront thinking and upfront investment is paying off a lot.

If you ask, *What is the return on investment?* it depends on the organization. Your return on investment is different than mine because my developer cost is different from another developer's cost. Or, my risk is lower than somebody else's. It depends on the risk, and it depends on the organization. So, definitely there is a lot of return on investment a lot. Like you said at the beginning, if you fix the one feature early, versus waiting at the end. If the organization has a three-week sprint, it's going to cost them a month versus getting it secured early on. So, if one developer is working, it depends on how much the developer is making, one feature is going to cost them a month as minimum delay, versus at the beginning. So, we are saving over that, and multiply with hundreds, multiply with thousand. That's a number, actually I can't say that.

Eileen: So, a big focus here at the SEI is transitioning knowledge from our research and from our client engagements to our stakeholders and the public. What kinds of resources are available now for folks who are interested in taking a DevOps journey?

Hasan: So, our role in SEI, we are practicing here in different environments and different technologies. And so, different organizational cultural things. When we learn, we are sharing our experiences, and through the SEI blog posts, and we are constantly going and talking at the conferences and sharing our experience and sharing what we learned in development communities. So, if you are in the restricted environment, how do you do that? If you're in a controlled environment, how do you do that? So that type of things we do. Through the SEI webinar series and blog posts and also the paper that we're writing and also talking through conferences. That's how we share our information with communities.

Eileen: Right, and your blog posts are available on the DevOps blog at insights.sie.cmu.edu/devops. I think that's where they are. So, what's next for your team for new work in DevOps?

Hasan: We are working right now to include more practices, and to expand more automation throughout security. Also, we are looking for more about the blueprint type of approach to DevOps. We are getting a lot of questions: *All right, I've heard of DevOps, and I would like to implement that in our organization and I heard how Amazon does it, how Google does it. Can I use the tools the same way that Amazon and Google do?* We are saying, *No, you cannot get the tools. There is a process. You have to understand who you are first, and what your organizational skillsets are.* We are working more about understanding the needs and defining DevOps based on the needs, not the goal of releasing quickly. What are the principles of DevOps that can benefit an organization, not only for taking the DevOps principle to a quicker-and-faster release, but at the same time thinking about *What are the principles I can use from DevOps for organizational effectiveness and traceability and other components?* So, we are working on that topic right now and giving communities more intelligent ways to implement DevOps.

So, we are saying maybe a blueprint approach or some sort of like a sharing typical reference architecture and typical abstract model about how the DevOps can be implemented in your organization, specifically for DoD-centric [environments] to develop more on the topics.

Eileen: And so to help people, to guide them toward the decisions that they need to make, with helping them consider all the factors that are in that context.

Hasan: Absolutely. So, being an FFRDC [federally funded research and development center]. This is the SEI, we are not looking for specific tools, and I don't believe that one tool is going to solve everything. When you look at the tools they think, *We'll solve everything in the DevOps*. No, first we have to understand *What are our needs? What do we have to do in organization? What our applications [technical]stack is required?* So, be intelligent in picking the right tools and putting procedurals things in the organization. We can develop those things. That's the work we are working right now, specifically.

Eileen: Great. That sounds really exciting, I can't wait to her what more comes out of it. Hasan, thank you for joining me today to talk about this work.

Hasan: Thanks for having me on.

Hasan: It was a really good time. Hassan recently co-authored a paper entitled *Where to Integrate Security Practices on a DevOps Platform*. That is available in the *International Journal of Secure Software Engineering*. He and his team regularly author posts on the SEI's DevOps blog at insights.sei.cmu.edu/devops. We'll provide links to all the resources we discussed today.

The transcript of this podcast will be available on the SEI website at sei.cmu.edu/podcasts. It will also be available on Carnegie Mellon University's iTunes U site and the SEI's YouTube channel. As always, if you have any questions please don't hesitate to email us at info@sei.cmu.edu. Thank you.