Achieving Continuous Authority to Operate (ATO)

Featuring Shane Ficorilli and 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: Good afternoon. My name is Suzanne Miller. I am a principal researcher here at the SEI. Today, I am joined with two colleagues that are actually in my own group, Shane Ficorilli, who is a software engineer in Continuous Deployment of Capability, and our boss, our mutual boss Hasan Yasar, who is the technical director of this directorate.

I want to welcome both of you. Today, we are going to be talking about authority to operate, commonly known as ATO, and some of the things that are happening that are changing the landscape of how we secure our systems and software systems in the DoD. Before we get started with that topic, tell me a little bit about yourselves. Tell our listeners a little bit about what brought you to the SEI and this kind of work. Shane, why don't we start with you.

Shane Ficorilli: I am a DevSecOps engineer on, like you said, the Continuous Deployment of Capability Team, and I specialize in DevSecOps pipeline architecture and engineering. So, one of my primary roles is working with our customers to help them make the right decisions whenever they are starting to actually implement their DevSecOps pipelines.

Suzanne: What did you do that led you into that kind of work?

Shane: I come from an information-systems background. So, the blend of infrastructure as code, being able to automate the creation of virtual infrastructure, was really the ultimate blend between being a sys admin and the development side as well. I found that perfect match.

Suzanne: Hasan, tell us a little bit about your journey to the SEI and what brought you here.

Hasan Yasar: It is a long journey.

Suzanne: I know, I just want you to say a little bit.

Hasan: I have been at SEI almost 10 years. I started in 2010 since I have been at SEI. Why I am here at SEI and CMU, I had a lot of industry experience, and I would like to bring the industrial experience to the different domains. Not really doing the work but helping by knowing the challenges in software delivery, software deployment, and making sure that you are able to deploy and deliver any capability that is timely. Also, with a security mindset as well. So, as part of the CERT now, the idea of DevOps started seven or six years ago. Now, it is getting more knowledgeable in terms of everybody wants to do it.

Now we have more challenges we have to do workwise, like, *What is the right way to do DevOps? What is the right way to do DevSecOps and ATO?* So, one reason I continue here is because I would like to help the communities. I am really happy to have Shane here as well. Before this, Shane was a security expert taking a class in the [Carnegie Mellon University] Heinz College. I said, *Shane, we have to do more application side of it because everybody is talking about the operational side, but we have to build security in while we are building the application*. That is the reason Shane is here, to build security in and get that experience, not really addressing the security problem after the fact. Let's address the security problem while we are building the systems, and ATO is one of the ways to do that.

Suzanne: Let's move into talking about authority to operate. So, this is for our DoD customers, this is a well-known attribute of any software system. I must have an authority to operate so that I can operate out in the field. This is a process that has a whole lot of early acquisition all the way through development, all the way into deployment kinds of activities. But, historically, it has been applied to a single system. So, if system A has an authority to operate that is a different ATO than system B, than system C. What kind of problems does that style of ATO engender in the kind of ecosystem that we have today?

Hasan: So, I can start a little bit. Before diving in to the problems, I would like to open up to what an ATO typically does. Almost every organization has to approve the software before putting [it in] the production environment regardless of where they are—maybe in healthcare, maybe finance, everybody has to do that. Specifically, in the context of the DoD, ATO is based on the Risk Management Framework that the organization has to follow.

Suzanne: The NIST [National Institute of Standards and Technology].

Hasan: Yes, NIST 800-37, which is the Risk Management Framework. They have to implement it. But how the information-assurance person authorizing the application [who] will be pushing it to the production environment, they are doing a manual process, which a checklist. What the checklist is based on is 800-53, which is the standards controls. It is more than 1,000 [security] controls. So, the organization has to pick the right controls based on what they are looking for,

application specific, and monitor every control before pushing the application into the production environment.

Suzanne: So, these securities controls are things that are different depending on the context of the application and the context of use. So, there is quite a bit of work in figuring out what are the correct controls that are applicable to the system and applicable to the system in the context it is going to be used in.

Hasan: Absolutely like picking [the] right controls and also the monitoring of which control has been implemented in the lifecycles, not in just the testing phases, but the requirements, or some organizational track-modeling concept. How you get the track modeling based on the controls through the right mitigation strategies? So, it is really a manual process. How you select it? How you monitor it, and also how you approve it. So, if there are not any things process-wise—automations or traceability—it becomes a bottleneck for multiple reasons. One, you cannot really deploy the application on time. That will take sometimes six months, maybe a year. Second, you are going to spend a lot of time and money redoing the work over and over again. And third, the most important one, if there is any vulnerability that has been discovered in the production environment, how much more quickly can you update the patch on time? Because if there is a delay in the approval process, then if you are not able to update the production environment in a timely way, it becomes vulnerable for adversaries to take advantage of it. So, [there are] a lot of challenges in that space.

Suzanne: These kinds of problems and one thing you did not mention is, *Oh and by the way, if I decide that I need to change the system, then I also have to go through this process over again. I have to review all the controls, make sure they are still applicable, make sure that my mitigations are still relevant, etc., etc., and then verify that they have actually*…so that is I think one of the things that in the communities I work with that idea that, *I have to freeze something and I cannot allow it to change even if my operational environment changes or my technology changes*. That is a real barrier to us getting our systems up to the speed of relevance, which is a theme a lot of people are talking about. I know the community at large has come up with this idea of a continuous ATO, continuous authority to operate. So, this is a different way of approaching that whole problem. Shane, why don't you describe for us what is it that is different about a continuous ATO from a traditional ATO.

Shane: So, traditionally as Hasan had said before, the approval process had often taken place at the very end of your software development lifecycle, right before you would actually deploy to either your staging or production, and that would be a manual process. So, with continuous ATO what we are trying to do is to have that process take place throughout the development lifecycle, specifically at the continuous-integration phase. Originally, that manual approval process really helped to reinforce the waterfall style of delivery, where with approving and checking against the

risk-management frameworks continuously as you are iteratively developing, helps to develop value at the speed of relevance like you had said, because it is happening as you are developing instead of at the very end before deployment.

Suzanne: So, it supports the iterative, agile development processes that are being emphasized in DoD right now.

Hasan: One of the challenges that we were talking about at the beginning is why it takes time manually because we have to go through all the dependencies of the things we are building. Dependencies, which is all the libraries, all the infrastructure pieces, have to be taken and looked at over and over again versus when we look at application delivery or the application-building process, we are writing a very small set of code. But we are spending a lot of time rechecking the back end that has been done already in the past.

Since we do not have traceability of back-end components, how much we looked at and how much we reviewed and approved, we go back to the manual again and again. So, what we are seeing with continuous, we have already done some of the preliminary dependencies that have been approved already, maybe some other dependents, maybe some services. So, just getting approval of only the core pieces, which is a very small piece, has been approved; but what other artifacts are generated artifacts to the pipeline like using continuous integration, continuous deploying pipeline, then will generate the right artifacts to the organization or information to this person who will approve only the changes that have been made, not everything else, that is the beauty of the continuous process.

Suzanne: So, there is an assumption here that we have a stable infrastructure and that we understand it well enough to instrument it so that we can determine if there have been changes to it. Those are some assumptions that we are making. These are assumptions that are actually pretty easy to fulfill in our cloud environments and in our modern kind of development environments. But in our modern development environments, it is actually feasible for us to do that. So that is one of the things that has shifted I would say in the last 10 years that makes this idea even feasible to consider.

Hasan: Yes, that is right.

Suzanne: When we think about continuous ATO and where it is now, where are you seeing this being applied that you can talk about, and what are some of the challenges to getting this idea accepted within communities for whom it would have relevance?

Hasan: One challenge, which is a typical challenge. I am sure you can dive in more on the technical side, Shane. The first one is cultural.

Suzanne: I knew you were going to say that. Hasan and I talk about cultural issues all the time, so this is a common conversation.

Hasan: It is a belief, right? The culture and security person or information person has to believe and trust the developer, so trust other components. We do not have that trust capability yet because there is kind of like a conflict between each other, because, *I do not know, and I do not trust you.*

Suzanne: *I did not do it myself.*

Hasan: *I did not do it myself and I have to look at it. I have to really touch it. I have to feel it*. It is a culture problem. You have to enable the issues to make sure that we are going to trust each other. It is a learning curve. It is like a journey, like sharing old artifacts. Building the trust relationship between the security expert and the developers. Building the trust relationship between the human and the machine itself is another thing. If the system is generating a lot of false positives, that person will be looking around at, *What I am going to decide*? Now if the person is taking a risk and approves the system, *If something happens, am I going to lose my job or if something happens to me, maybe it is going to cause so much other problems in the organization*. What will happen? So, it is creating a fear. Now the fear is not sometimes a bad case but sometimes it causes [fears] like, *What I am going to do next*? which is blocking. So, it is all creating a cascading effect, not trust, creating a fear and creating problems. It is all awareness at the cultural level.

Suzanne: There is an aspect as well that people tend to self-select into roles that they are comfortable with. There is a risk-averse aspect to security. Security is the epitome of risk aversion.

Hasan: That is the fundamental definition.

Shane: *How do I avoid risk?* is what my security is about. So, the people that are going to self-select into that role are not naturally going to be trusting. We have to do more than might be typical with just a developer to actually engender that trust. So, what are some of the aspects of the continuous ATO process that have evolved to actually help with that?

Hasan: The first thing that I did for the culture elements, I brought the security expert and the developer into the same room and let them talk. Share their ideas or share the burden, share the problems that they see, building the trust at the human level. So, in the end the developer generates some artifacts and the IA [information assurance] person will trust the artifacts because they see they are in the same boat, and they have the same goals, and they are sharing, which is being in the same room. Then the tooling aspect, which Shane is going to cover the problem

from the more tooling side of it, *How to do it?* And, it also depends on the tooling process as well, like in the infrastructure pieces of the deployment pipeline that can be done.

Suzanne: Shane, what are some of the things that have evolved as strategies that help the risk-averse security community trust that the infrastructure is going to remain less vulnerable, that the applications are going to get the attention they deserve? How does the tooling side help with that?

Shane: I would say that one of the biggest factors that really allow for continuous ATO to take place is the creation of all of your environments through infrastructure as code. That really helps both people on the operation side, the security side, and developers to get an environment in a scripted way that can be approved as code. Then you are actually able to say, *Not only are these specific environments approved for deployment, but we know exactly what versions of software are going to be in those environments because they are defined and stored in source control*.

Suzanne: For those of our listeners that are not familiar with that particular concept, give us 50 seconds—not 52, not 55, but 50 seconds—on what is infrastructure as code.

Shane: Infrastructure as code is basically different types of scripted languages that allow for the virtual creation of infrastructure across various platforms. You can create anything from containers to virtual machines to any type of infrastructure in any cloud provider, such as AWS [Amazon Web Services] or Azure. Pretty much any type of virtual infrastructure that normally would be created in a manual process, you can write scripts as code and version control those to be able to re-create environments on demand.

Suzanne: So that control, that version control, that assuring of, *I am not going to be able to change this or that you aren't going to be able to change this just because you feel like it*, that is one of the things that gives the security community a little bit more faith that this is a safe thing to do.

Shane: Absolutely.

Hasan: In an infrastructure, it is not just the operating systems, it is also what tools or what libraries do the required application run or the application dependencies or the application configuration are also part of infrastructure code pieces, too.

Suzanne: My understanding is that you also eliminate things that are not needed. So, one of the big issues in the security community is, we fondly call dead code, code that is just there and is not doing anything and creates a vulnerability just because of its existence and possibility that it can interact with other elements of infrastructure. So, this infrastructure as code actually eliminates one of those risks in many environments.

Hasan: Eliminate and also, it is going to create traceability. It is also going to create inventory mechanisms. It is going to create transparency so that going back to the cultural trouble we talked about at the beginning, the IA person knows exactly what the developer needs to write, what a developer needs to run [as] an application, which is in the code itself—not getting a piece of paper saying, *Here [are] my dependencies* and then follow it, and it is going to run an environment. No, instead say, *Here is my script. Run it.* Then the IA person exactly knows that application is calling here and there and connecting this dot, connecting here, downloading that application or dependencies. They have full visibility. That is how to trust develops from visibility at the beginning.

Suzanne: Transparency is one of the big things.

The other thing that you haven't mentioned, though, is that the speed at which I can repeat this process is so different than provisioning infrastructure from a manual viewpoint. If for some reason the security person has some doubt about whether this version was provisioned correctly, they can just run it again and do compares and things like that. So, that ability to pretty much on-demand re-provision and restart from scratch, that is something that in a manual environment…

Shane: It could never take place.

Suzanne: Oh my gosh, you would be here all weekend. People won't do that.

Hasan: That is kind of like assurances for risk-aware person would be guaranteed that I have a fully automated and fully repeatable environment. If something happens, just changing one of the libraries and repush again, so which is a guarantee to the person that there is a risk probably, but if something happens, I am able to change infrastructure any time if I need to.

Suzanne: So that resilience of being able to recover from, and not just from the viewpoint of if something does not feel, kind of, feel right, it is also from the viewpoint of if a vulnerability is identified, I can swap the vulnerable element out for something that I have trust in and continue operating. This has huge implications for our operational environments and our ability to deploy in speed of relevance. When we have changes in operational environments and technology, we need to pivot, areas like that. Are there any particular success stories that you want to point to that help people to understand just how much effect this has had on the community?

Hasan: I would respond in a different way, Suzanne. Instead of saying success stories, I will give them examples, they do not have right mindset, infrastructure component, or not a continuous ATO, and they are failing miserably.

Suzanne: Okay so what are the failure modes that if you do not do this you can end up?

Hasan: If you go to any of the security news pages or any type of news organizations you will see a lot of data breaches happening. These people are not implementing proper infrastructure behind the scenes. Equifax is one of them. Or many other data breaches [are from] not having the right infrastructure management or not dependency management on it.

In the AWS case, it happened recently. In Capital One it happened recently. There are many examples where it is not properly vetted in the process and not able to keep trace of any key management in the code itself. Not able to manage the software-delivery deployment pipeline, who is touching, who is changing, or not able to find out all the dependencies in their build process, kind of like ad hoc developer pulling here and there. Even some organization they do not know anything running in the infrastructure, [as in the] OpenSSL example, like OpenSSL had a big problem [Heartlbeed] arise around 2013. A lot of organizations had no idea whether they were using OpenSSL or not. So, there are bad stories because we do not hear the good stories because they have been great. So bad ones are affecting our life.

Suzanne: This is actually a call to say continuous ATO is something that commercial organizations, even if you are not subject to the same kind of regulations like we are in DoD, this is something you should be thinking about.

Hasan: I completely agree. When we looked at this at the beginning, approval depends on which organization you are in. If organizations are in the healthcare industry, they have to follow up to HIPAA requirements, which requires that certain control has to be implemented. Financial sector has Sarbanes-Oxley. They have to implement [those] controls. Further, look at the GDPR [Global Data Protection Regulation] requirements, we have to do similar things, finding out how we are handling the data. Somebody has to be in approval process. In the DoD context we are saying ATO, but industry is doing similar things. But the foundational technology, which is DevSecOps or DevOps, is helping to achieve the goals of building the primary infrastructure, building automation.

Suzanne: So, we have DevOps, which is now a very familiar term to everyone, but you have just starting talking about, and we have been talking at the SEI about, DevSecOps. Talk for a minute about how does DevSecOps relate to this continuous ATO process?

Shane: I would say that DevSecOps is really just DevOps done correctly.

Suzanne: Now you are throwing down the gauntlet.

Shane: With DevOps you are really automating your deployment and delivery pipeline. You are taking advantage of a lot of the automated testing that takes place within the continuous-integration phase. But with DevSecOps, you are now taking advantage of all of the security automation that you can incorporate throughout that pipeline.

Anything from automating static code analysis to automating the checking of vulnerabilities in your third-party dependencies to doing automated container-image scans or automated dynamic-analysis testing. So, you are basically just adding that additional layer of security automation into your deployment pipeline, which is something that should have been automated from the beginning anyway, so that is why I said it is really just DevOps done correctly.

Hasan: That is true, but organizations are not feeling it is as necessary to use security activities. It is mostly not intentional, but [if] it is not bringing money to organizations of value, [they] forget or ignore security in the lifecycle.

Suzanne: It is what we sometimes call a hygiene activity. It is, *Make sure you remember to brush your teeth*, but it is not something that is going to make you look better. It is not like wearing a new suit or anything.

Hasan: We have to call out [that] security clearly has to be part of it. It has to be part of throughout the lifecycle, not only having static analysis, not just having dynamic analysis. It has to be throughout the lifecycle. Really from the beginning all the way to the end, that is the way we can achieve the continuous ATO. That is the way we can achieve selecting the right controls based on my requirements and implementing during the lifecycle and trace that the requirement has been properly implemented in the lifecycle, which is very connected to lifecycle, which activities [are] part of the overall ATO process.

Suzanne: Anything that we haven not talked about that you wanted to make sure our listeners know about continuous ATO and DevSecOps, what things are going on here at the SEI?

Hasan: A couple of things I would like to mention as well with DevSecOps and ATO. The continuous ATO, that can be achieved through DevSecOps only. If you or your organization are looking forward to implement continuous ATO, they have to have a great DevOps or DevSecOps environment in terms of the traceability pipeline. Then all the stakeholders…which we have a diagram actually through the CERT SEI website that can be used as a map to lay out what needs to be done in which state [and] what you have to do. We do not want to go over every step, but I really encourage people to take a look and then download all the posters. Then think about building up the pipeline. Get all the stakeholders together, that is how we can achieve the cultural issues and enable that.

Also building a right process into their environment, like how are we going to build up incremental, iterative development? If we are building a great environment, then having an ATO just in [environment] pieces, but if we are building a bigger monolithic application in the pipeline, we are not going to have continuous ATO. We would like to have a quick build based on dependencies. So, at the SEI we have been helping the DoD build up all of the right

containerization concepts, then DoD or anybody will be able to pull up the hardened containers, putting [in] their application [that] has been authorized and approved already.

Suzanne: This idea is going beyond an individual infrastructure. That is the thing that we are seeing is looking at this from a DoD enterprise viewpoint of saying, *Everybody needs hardened containers.* What are the things everybody needs no matter what you are doing, and where you are looking for these sort of, customization being the exception rather than the rule, is one of the ways that we are able to help people to move through this more quickly. That is important, and then this idea of culture, but the other aspect that I get into when I talk to people about this is governance. And that is an aspect not just of your culture but of your process of, how do we make the decisions that something is OK, something is not OK? Who makes those decisions? Those are some of the stakeholders you are talking about. And making sure that the people who can say *no* are part of the process of understanding how do we say *yes*. That is one of the areas that the SEI is a little bit more conscious of when we go in to talk to organizations than some of your DevOps tool vendors because we know that, *If you can say no, what makes you comfortable saying yes,* and those are some of the things that I know we have run into in some of the customers that we have worked with.

So what is in the future? What are the things coming up that you are just itching to get into that—we are still dealing with other things, but what are some of the future things that you think are going to be coming along with continuous ATOs and DevSecOps? What do you want to fix?

Hasan: Shane, do you want to chime in?

Suzanne: What is left out there for you to fix, Shane?

Shane: So, I think the idea of using hardened container images from the get-go is really starting to catch on. I think the fact that the DoD is taking it upon themselves to provide a repository or registry of hardened images for all of the different services to use is really going to help to solve a lot of issues for vulnerable dependency usage within your infrastructure right away. I would like to see the community adopting that, in general, using hardened containers, but it is difficult to create basically a worldwide registry of hardened containers. It is a lot more difficult than it may seem.

Suzanne: Well, we have got legacy systems to deal with, and that adds a wrench into this whole works. There are things that you look at and you go, *Not really worth building a container around, but, at the same time, I am not sure that I want to let it out in the wild in its own state*. So, there is some of that that I know is something that we deal with on a regular basis.

Shane: And I know one of the other things that we have seen, too, is that within DoD, a lot of organizations are really trying to tackle the challenge of decomposing their huge monolith

applications into microservices. So, using something like a strangler pattern or kind of chiseling away at their big monolith, one service at a time. It has been really cool to see them take on that challenge.

Suzanne: There has been an evolution in processing. When I was developing stuff in the 80s, we had so many resource constraints that an awful lot of these systems from that era are very tightly coupled because we just didn't have room to have external APIs. We did not have enough computing resource, either storage or memory, and so we did a lot to get the performance we needed. Well, now we are reaping the effects of that because now we have all these things that are intertwined like spaghetti, and we have to disentangle them so that we can actually understand… We have room now. So how do we actually make it so that we can see what is there and deal with things—*This needs to be containerized, this needs to go away altogether.* I mean, that is a whole set of challenges that I know many of my customers are dealing with right now.

Hasan: Another challenge is that we would like to work with specifically more hardware-based systems. If we look at the current tooling schema, it is more about event applications, but we would like to spend more time building the right process, the right concept with the right tooling that will help the organization build up embedded systems or the hardware pieces or building the firmware on it. *How can we build up modularity in the firmware level or the hardware level so we have a continuous process, continuous-integration delivery?* I call it continuous security that will help us to build up a continuous ATO overall process. That is another exciting area where we would like to continue working and exploring ideas.

Suzanne: You are going to busy for a while. Nobody gets to retire in the near term.

Hasan: There is a lot of work to do for everybody.

Suzanne: There is, there is.

Hasan: We have a lot of challenges: complexity, dependency. We are living in a very connected world. We have a lot of hardware, which depends on the software. Now we have different layer of applications, different layer of tools. It is just getting so much, I do not want to say problem, more about complexity, and we have to solve it.

Suzanne: We are moving from complicated to complex. *Complicated* where we know what the scope of the problem is, and so it may be difficult to manage through it, but we know what it is. *Complex* sometimes we do not even know what the scope of the problem is. We can't see all the elements, and that is I think what you are referring to, we cannot see everything the way we used to, and so we have to find new ways to figure out how to deal with that.

Hasan: We are living in a dependent world right now with very basic web application. If you look at it, now there are more than 25 dependencies as layered and building a single web application versus 30 years ago, we had HelloWorld! in C++…

Suzanne: We had the seven-layer OSI model. Seven seemed like so many at that time.

Hasan: Now in the application, many frameworks just to get on, many framework dependencies we do not know what is really behind it. If you do not know what that is going to be like, if you do not estimate what are the complexities, what the dependencies are.

Suzanne: Well, I want to thank both of you for joining us today in talking about this important issue. I think not only are the military folks nodding their heads up and down, I hope we got the attention of some of the folks out in the commercial industry as well. We do have transcripts that will be available for this that will include links to resources. There are a lot of resources related to this, both blog posts, other podcasts. We want to make sure, courses. Those will be attached to the podcast, and I thank all of our listeners for viewing this today and thank you again for joining us.

Hasan: Thank you for having us.

Shane: Thanks for having us.

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.