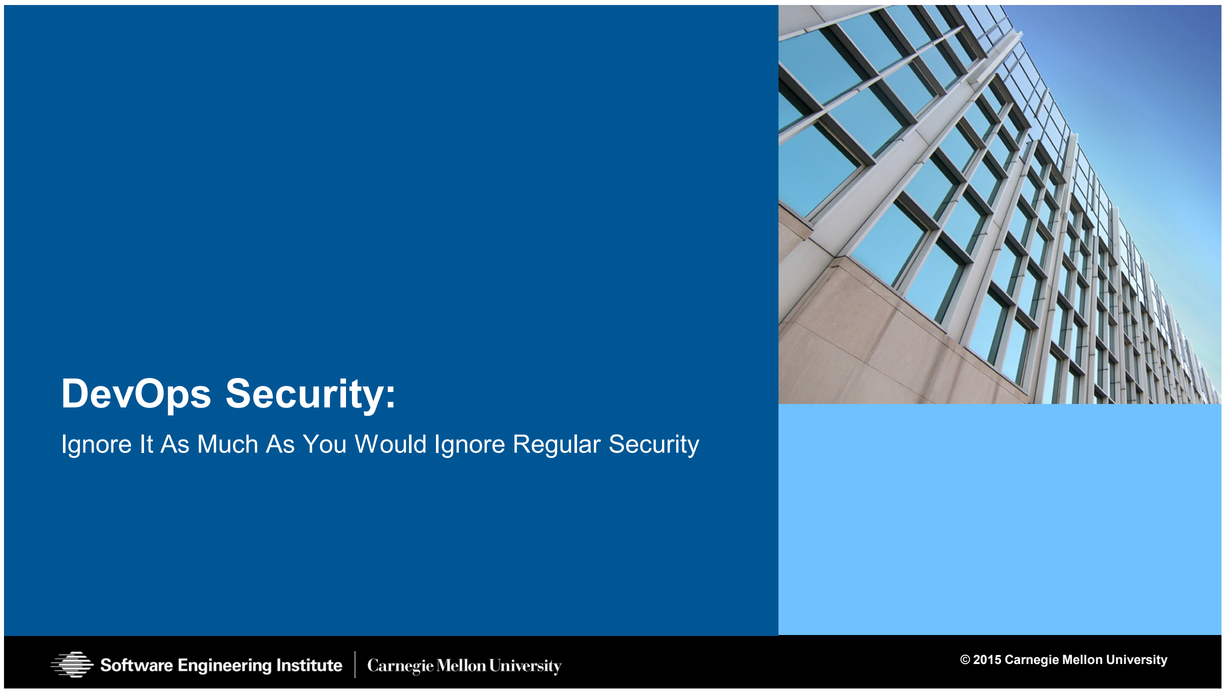
SEI_Webinar_DevOps_3

Table of Contents

DevOps Security:



DevOps Security:

Ignore It As Much As You Would Ignore Regular Security

Software Engineering Institute | Carnegie Mellon University

© 2015 Carnegie Mellon University

**003 Shane: And hello from the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania. We welcome you to the SEI webinar series. Our presentation today is DevOps Security: Ignore it as much as you would ignore regular security. Depending on your location, we wish you a good morning, good afternoon, or good evening.

My name is Shane McGraw. I'll be your moderator for today's presentation. I'd like to thank you for attending. We want to make today as interactive as possible. So, we'll take questions throughout the presentation and again at the end of the presentation. So, you can submit any questions to our event staff at any point. You'll see a questions tab on your event console. Simply, type in your question, click send, and we'll address as many as we can throughout the presentation.

We'll also ask a few polling questions throughout today's presentation. And they will appear as a pop-up window on your screen. In fact, the first polling question we want to ask today is how did you hear of today's event. And you'll see that pop up now.

Another three tabs I'd like to point out are the files, Twitter, and survey tabs. The files tab has a PDF copy of all the presentation slides today along with other resources from CERT and their work on DevOps. Lastly, the survey tab we ask that you fill out at the end of the presentation as your feedback is always greatly appreciated. For those of you using Twitter, be sure to follow @seinews. And today's hashtag is #seiwebinar. Once again, you'll want to follow @seinews and the hashtag is #seiwebinar.

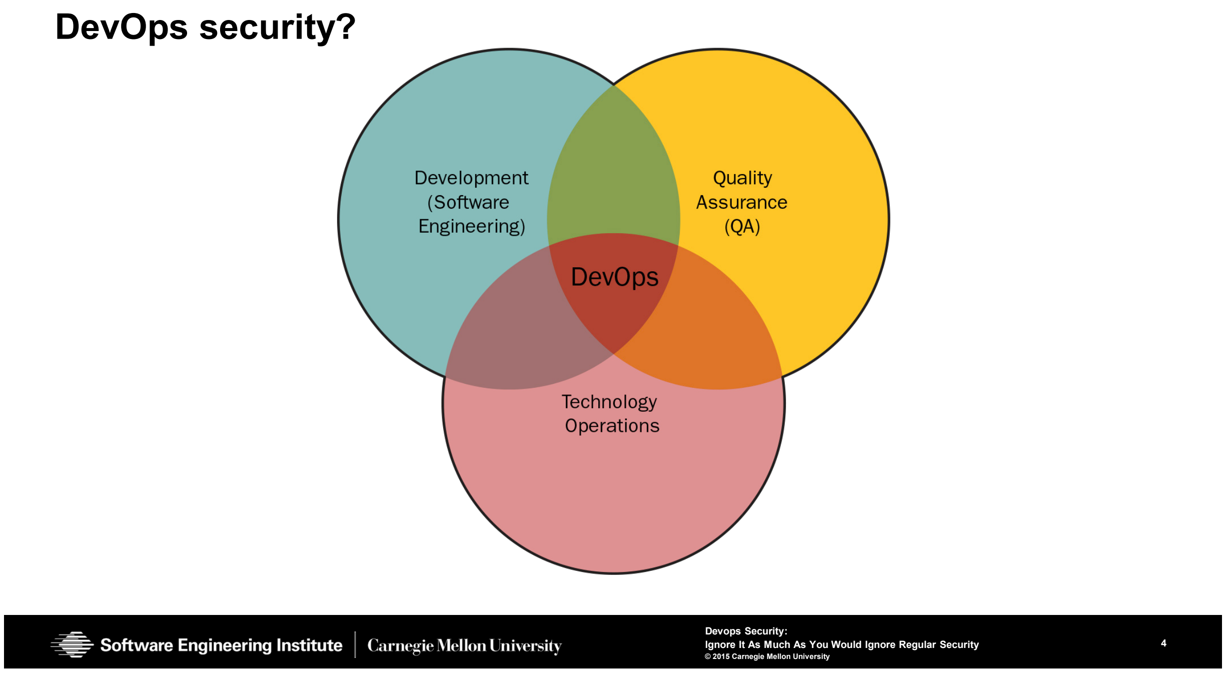
Now, I'd like to introduce our presenters for today. Timothy Palco is a software engineer and developer with a background in a variety of languages and tools starting with a Commodore 64 in 1987. His experiences working in different cultures from start up to enterprise have helped him focus on improving the development process. Since joining CERT full time in 2012, Tim has designed and built tools to support on-demand virtualized machines, network monitoring and analysis, and wireless sensor monitoring.

Christopher Tashner worked for a U.S. Department of Defense agency for over five years before joining CERT in 2006. His area of specialty and training include cryptanalysis, reverse engineering, software development, web application development, vulnerability analysis, and digital forensics. And now I'm going to turn it over to Tim Palco. Tim, welcome. All yours.

Timothy: Thanks. Thanks Shane. Hi, welcome. You might notice the title of the presentation today is a little bit of a joke about ignoring security as much as you would ignore regular security in the DevOps space. And it's kind of a joke because security is often one of the first things that's kind of shuttled off the development process after you get through testing and all the time that you try to build in for release and all of that. But the point I guess I was trying to make with the title is that with DevOps you can't ignore security any more than you would on any other type of space.

And it's not to say that we ignore security all the time. We always pay attention to the things like authentication and permissions issues, SQL injection and the like. But in DevOps, there's so much energy going towards the chains that it requires to transition your team to be using new technologies. And you're kind of changing your culture. It's definitely easy to forget about security. So, what we're going to talk about today is really security in this kind of specific to DevOps.

DevOps security?



DevOps security?

Software Engineering Institute | Carnegie Mellon University

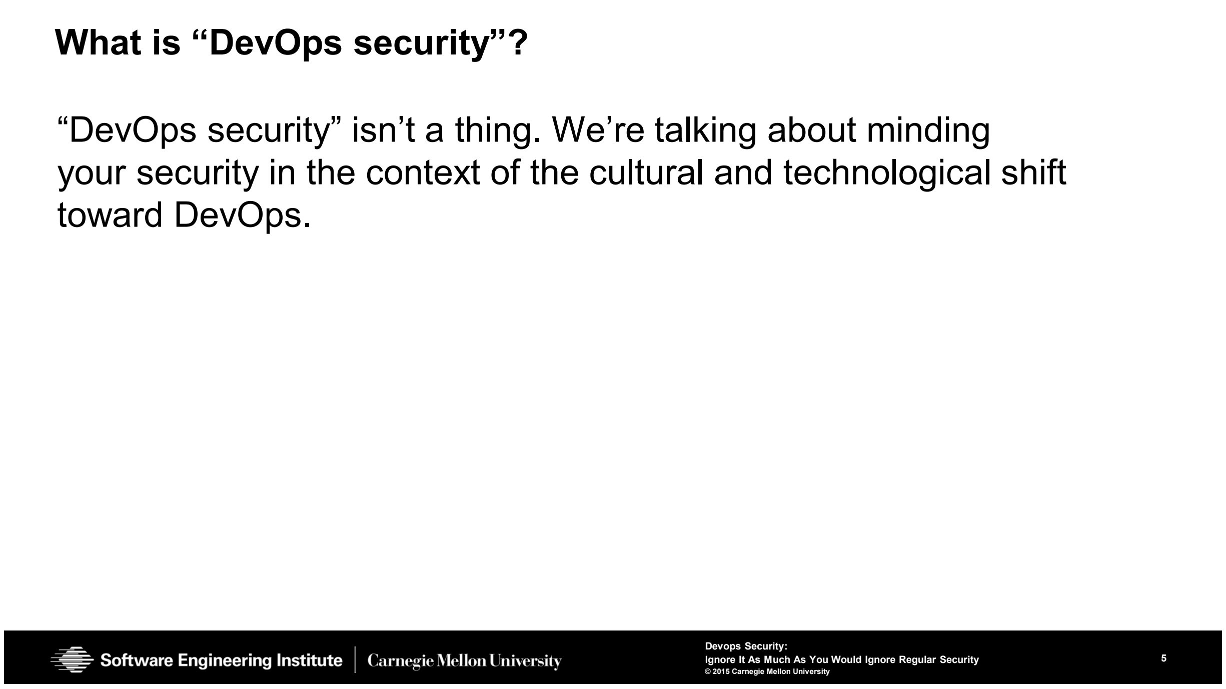
Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

4

**004 So, this graph kind of shows how a common understanding of what DevOps is and maybe a quick introduction for people who may not be familiar with DevOps or the concepts. DevOps is really-- it's not about merging these three groups together as the diagram might have you believe. But it's more about addressing a problem where in a lot of spaces you might a development team that kind of on waterfall approach will do their design and development and then pass it off to QA who does testing. And maybe they go back and forth a little bit. And then when they're ready and the software's ready, then they pass it to the operations people. And that's more of a waterfall approach. And that's how you would experience these kinds of things in a siloed environment.

But that's not really how DevOps intends it to happen. What we're looking to do is more pull these teams together so they're working together from day on rather than in that waterfall kind of approach. So, that's really the cultural aspect of DevOps. And then of course you have all the automation technology that's associated with DevOps, continuous integration and continuous deployment. Today we're going to be talking a little bit about some of the technologies associated with DevOps and also some cultural aspects as well.

What is " DevOps security"?



What is "DevOps security"?

"DevOps security" isn't a thing. We're talking about minding your security in the context of the cultural and technological shift toward DevOps.

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

5

**005 And the term DevOps security, I actually Googled DevOps security just recently just to see what turned up. DevOps security isn't really a thing. It's not a job. It's not a role. It's not maybe something that you would include as part of a software development project. But what we're talking about really is just security in the context of DevOps. So, security has a lot of roles. Security has its place in the software development project. But what we're going to focus on today is security specific to what DevOps talks about.

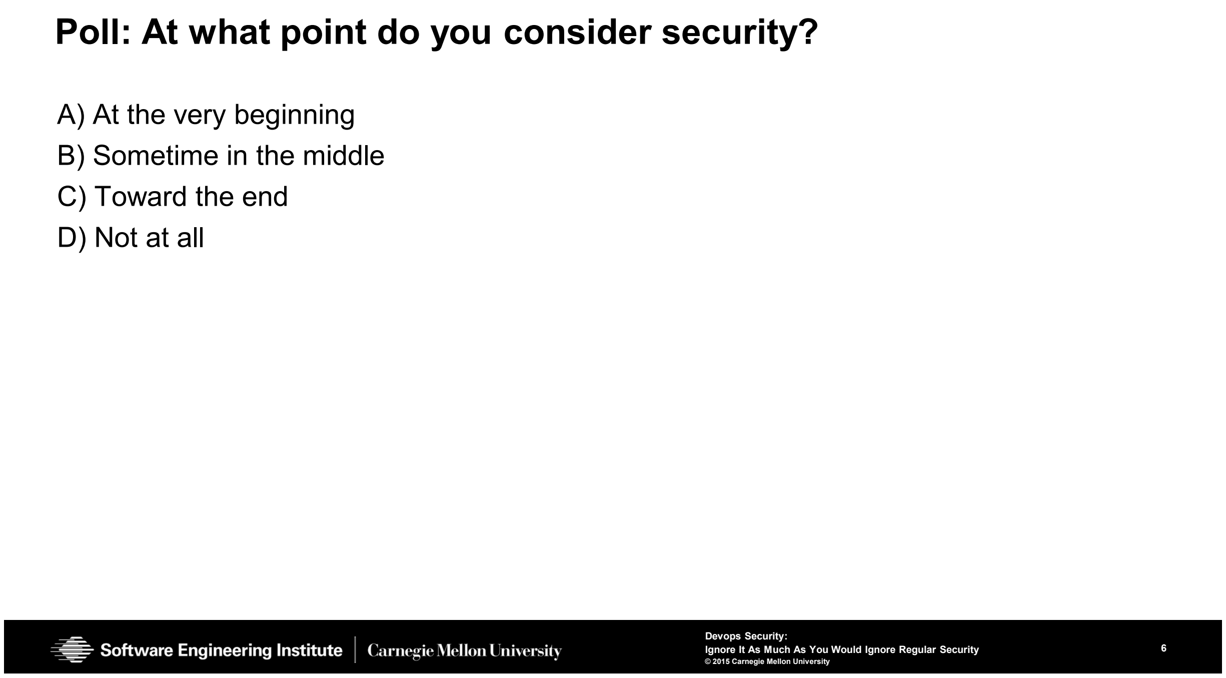
Christopher: Right. What we're talking about really is what are your quality attributes when you're going into your software project. If your organization is really concerned about security, that's sort of where what we're about to talk about comes in, right?

Timothy: Right.

Christopher: So--

Timothy: Right.

Poll: At what point do you consider security?



Poll: At what point do you consider security?

A) At the very beginning
B) Sometime in the middle
C) Toward the end
D) Not at all

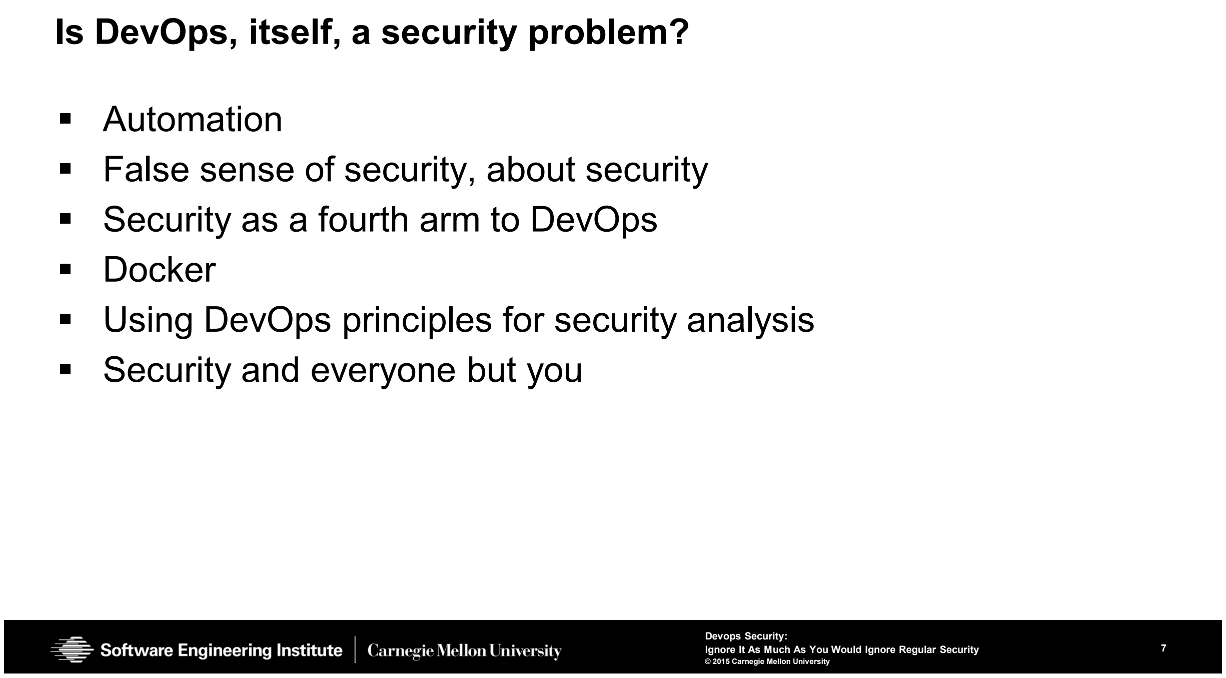
Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

6

**006 Shane: Okay, So, We have a quick polling question we're going to pose now just to get an idea of where you're at in the audience. And that question is at what point do you as an organization consider security, at the very beginning, sometime in the middle, toward the end, or D not at all. So, we'll give you about fifteen or twenty seconds to vote. We'll let these guys carry on. And we'll share the results in about another half minute or so.

Is DevOps, itself, a security problem?



Is DevOps, itself, a security problem?

- Automation
- False sense of security, about security
- Security as a fourth arm to DevOps
- Docker
- Using DevOps principles for security analysis
- Security and everyone but you

Software Engineering Institute | Carnegie Mellon University

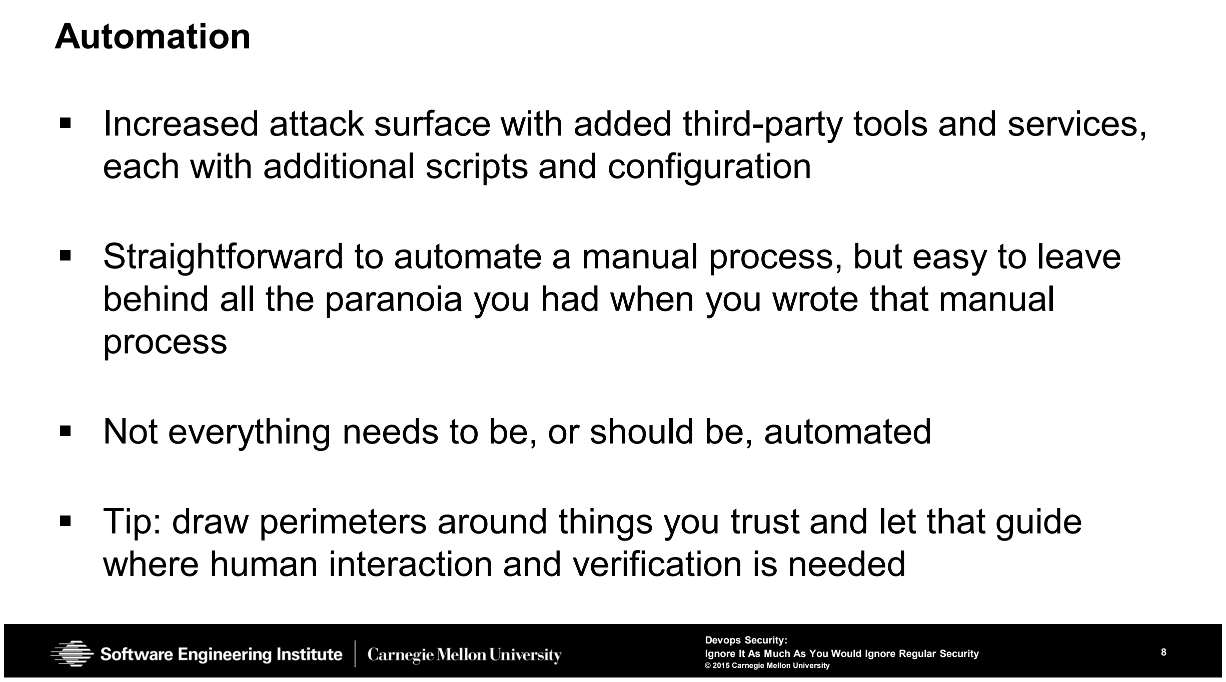
Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

7

**007 Timothy: Okay. So, these are the main topics we'll cover today. We're going to talk a little bit about, like I said before, the technologies associated with DevOps, Automation and Docker being those two, and see how we can apply security, or how security relates to these topics, and then some other things about security analysis and more cultural aspects of DevOps. And I don't know. We'll get into it.

Christopher: Yeah. Okay.

Automation



Automation

- Increased attack surface with added third-party tools and services, each with additional scripts and configuration
- Straightforward to automate a manual process, but easy to leave behind all the paranoia you had when you wrote that manual process
- Not everything needs to be, or should be, automated
- Tip: draw perimeters around things you trust and let that guide where human interaction and verification is needed

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

8

**008 Shane: So, we have our polling question results here, sixty-six percent at the very beginning, twenty-eight percent in the middle, five percent towards the end, and two percent not at all. So, I'm imagining you were hoping most people were at the beginning. That's where it needs to be addressed.

Timothy: It's good people are being honest.

Shane: Yes. And there are some honest people out there.

Christopher: Yeah, that's great. That's good to hear. So, let's start off-- when we start off thinking about DevOps, one of the first things we really talk about is infrastructure is code. Or, one of those key principles of DevOps is infrastructure is code.

And really what we're talking about is automaton. We're talking about taking as many of those processes to stand up your infrastructure and to deploy to your infrastructure, and automating them out so that we don't rely on individuals. And we don't rely on, "Oh, did I do every step in the process like I ought to have?"

Timothy: And varying stages of manual, like sometimes it's really manual command lines. Sometimes, you might have scripts in place. But still there's a lot of human interaction. And it's about moving toward being more fully automated.

Christopher: Right. And when people think about automation, one of the first things everybody seems to think about is speed. If I automate it, it's going to make it faster. And it's not always necessarily the case.

But when we talk about automation, especially in the context of security, what we're really talking about is control. We want to make sure that the processes that we're automating, that we have absolute control over those processes for reasons of now we know exactly what's going to happen. And we can audit that.

Timothy: Predictability too. You have expectations and it's tested.

Christopher: Right. So with that in mind there's still some issues with-- security issues with automation. With infrastructure is code, we're adding a lot of stuff here. We're adding a lot of extra code. And as the old saying goes, "All code has bugs." And so, we're adding code. So, in theory, we're adding a lot of bugs here.

Timothy: Yeah, a lot of code, and potentially because we're taking every single aspect of what will be your infrastructure, and networking aspects, and configuring firewalls, and configuration files for different services. And we're translating that all into the scripts, basically. And we're also adding third party code that's not something that you wrote, so say for instance, Chef or Puppet or deploying Docker containers. And we'll talk about that too. But all this third party code that you didn't write at all, and you're relying on that code to be secure.

Christopher: Yeah. So, this is all sorts of extra stuff that you have to make sure that those third party things are patched and updated as often as possible, and make sure that your code doesn't introduce any additional areas that an attacker or somebody malicious might take advantage of.

Timothy: Yeah, because this automation is additional programs running. And every program that runs is another thing that could be attacked.

Christopher: And with DevOps, even though we are trying to take out some of the manual processes, or a lot of the manual processes, not everything needs to be or should be automated. In some places, you want to have eyes on code, or eyes on the results of automation, or something like that. You want to have some gates in there in your continuous deployment, or your continuous integration cycle, where there's a little bit of sanity check. Not all automation tools are going to be able to-- or most automation tools are not going to be able to do every last thing and make sure that every last check actually performed properly.

Timothy: Yeah, the system isn't perfect yet, basically.

Christopher: Right.

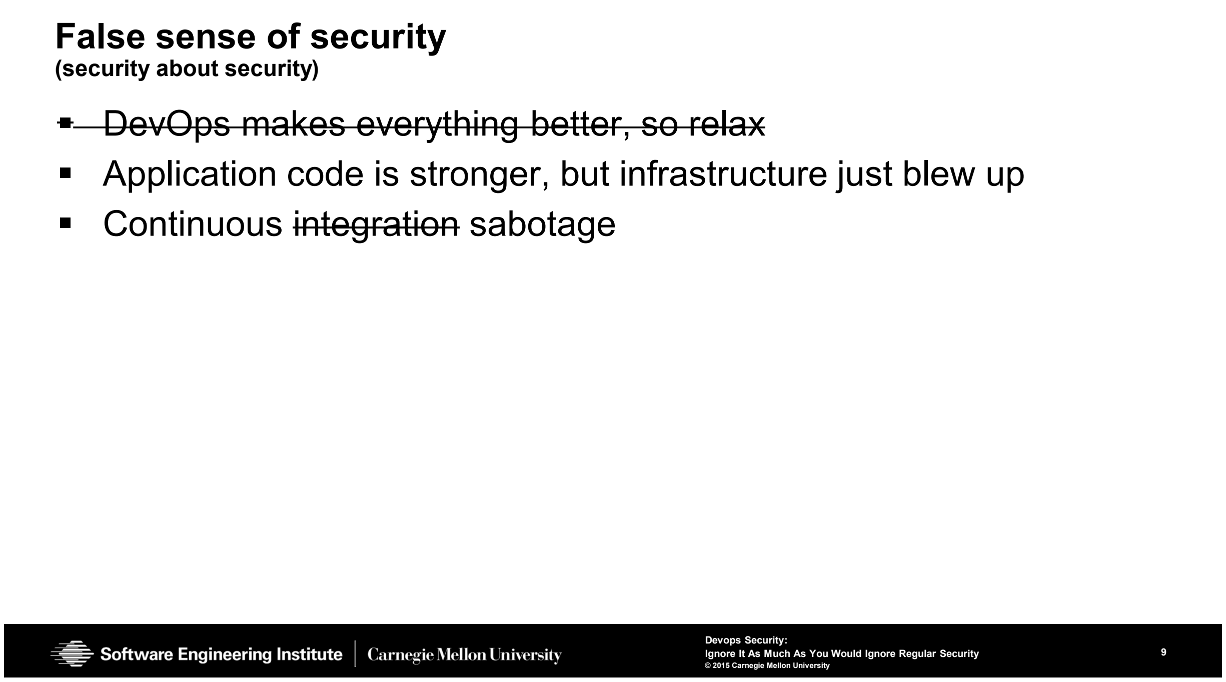
Timothy: Nor should we expect that it ever will be. And if someone claims that they have this perfect system, then be wary. It's probably not true.

Christopher: Yeah. Yeah so, you should be drawing parameters around the things that you trust and really looking at your system and making sure that the automation that you have in place, or that you're going to be putting in place, makes the most sense for what you're doing.

Timothy: Right. Right. To the second point in the slide that is whatever paranoia you had when you first wrote those shell scripts that you might be using today, you have to maintain that sense of paranoia. You can't relax just because you're using someone else's tool that their tool is well tested. But you still have to be paranoid about it.

Christopher: Right.

False sense of security (security about security)



False sense of security
(security about security)

- ~~DevOps makes everything better, so relax~~
- Application code is stronger, but infrastructure just blew up
- Continuous ~~integration~~ sabotage

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

9

**009 Timothy: The second topic is the false sense of security that you have about DevOps. There's a lot of hype about DevOps. If you've been to a DevOps conference, there's a lot of energy in a room of how great it is and what problems it can solve for you. But the thing to remember is that DevOps is not a cure all. DevOps can help improve your process. DevOps can help make things faster. And it can do a lot for you. But it could bring your team closer together. It could help you ship a better product. But the thing to not do is get comfortable just because you're using this thing that's supposedly making everything better. So, do not relax.

And while your application code becomes stronger because you may be doing more code reviews, and you may be testing your code more often, and you may have more eyes on your code, and you may be talking about all these problems more often, in the application space, you have to understand, with the automation, with continuous integration and continuous deployment, your backend, your infrastructure, all the scripts that are provisioning your servers, that whole space just got huge. So, like Chris was saying earlier, we have all these additional tools and scripts that are running to create your environment. So, well it's really just a point we already made. You really just need to pay attention to that because you have application code but now you also have code for infrastructure.

Christopher: Yeah. And like you say, DevOps is going to make your application better just by getting more eyes on the code and all that stuff. And from that aspect, in theory, it is going to get better in a security sense. But you have to keep in your mind and keep at the front of your project, eyes on what attributes do you care about, what does your business care about, what sorts of quality attributes are you interested in. And if security is one of them, what kind of security are you interested in? What's acceptable and what's not acceptable for your application? And design and build towards that and not just buy into the perception of DevOps being the magic bullet.

Timothy: Right. Right.

Christopher: And with things like continuous integration, there's other problems that can arise with DevOps, things that you might not really even consider when you're first putting these into place. And one of those we have up there, continuous integration, integration crossed out and continuous sabotage. The idea being that something as innocuous as a continuous integration cycle, which is going to help you in a situation where you have a security flaw, continuous integration and continuous deployment are going to help you get that security patch out quickly and relatively painlessly. But at the same time, if you have an insider threat, if you have someone that's disgruntled, or that wants to make some trouble, or make some money off of your application that's working inside, they might be able to quickly and easily, if your continuous integration is a push of a button that only a couple of people have to deal with, or one person has to deal with, then that hole can get out into your application pretty quickly.

Timothy: Yeah, once everything is automated to a point, and you get comfortable with it, which you shouldn't do, somebody makes a change like that. And all of a sudden for weeks, if you don't pay attention to it you could be very vulnerable.

Christopher: Right. This might be a good time to see if we have any questions.

Shane: Yes, absolutely. And we've got a good one in the queue here from Steve wanting to know some senior management CEO/CTO can sometimes feel removed from the technical day to day operations and are hesitant to implement DevOps due to the perceived risk of giving up too much unsupervised and unaudited power to people they don't directly trust. How can DevOps security mitigate these concerns?

Timothy: That's an interesting question. That's heavily based in the culture of a team really. And I think if I understand the question correctly, there's concern that DevOps will somehow remove power, I think, maybe from--

Shane: Maybe give the coder too much power on that code. Is there a way to mitigate that risk?

Timothy: Actually, that's kind of a problem that DevOps itself solves if implemented correctly because what we're doing is DevOps is really all about breaking down silos. So, you have teams working in silos. And instead of having walls put up for communication, you now have free flowing communication. And this concept of-- and we didn't bring it up yet, did we? Shifting left.

Christopher: Not yet.

Timothy: Kind of, but we kind of skipped over that. But this idea that from day one all of the teams working on a software development project are working together at the same table from the first day of the project. So, the idea being that if you're concerned that there's a particular team or particular people who might run with too much momentum, the idea is that really shouldn't happen at all because people are talking and communicating from day one. And those people, the management, would be sitting at the table also.

Christopher: And there's checks and balances. You can see what--

Timothy: Yeah. It's all a conversation from day one. So, that's not even so much a DevOps security question so much as it is just about properly implementing DevOps.

Christopher: Right. The other thing too, we mentioned earlier how the automation, it's a good idea to put in gates, to put in-- when you're adding your automation, to put in some stops where the code doesn't just automatically fly out depending if any given developer decides to deploy, that there is some sanity check involved. At that point too, you can have multiple people have to see it or something like that where it's not just this one developer has all the power or much more power to push something out into the world than otherwise without that continuous integration and deployment automation added to your application. Now, you have all this power given to lowly developers or some guy that's only been there for--

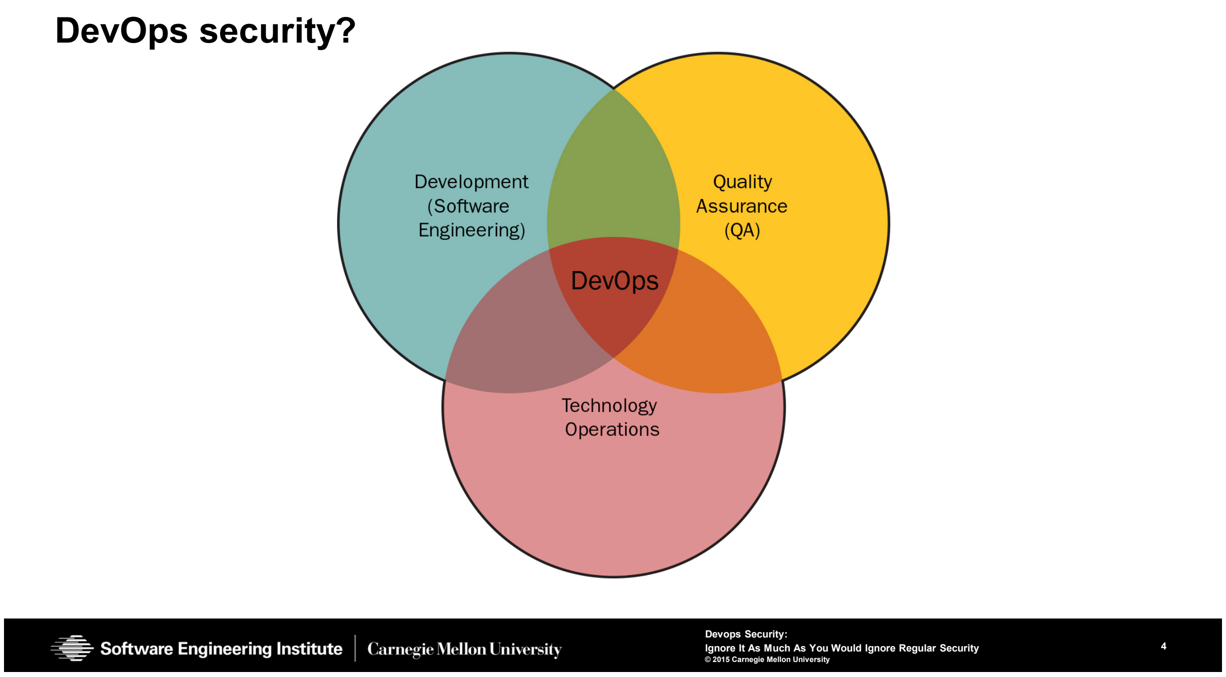
or a contractor or something like that. Now, you can add that somebody wants to deploy, what are they trying to deploy in there check.

Timothy: Right. Everyone's eyes are on the problem.

Christopher: Right.

Timothy: Also, I think maybe another interpretation, not to spend too much time on it, is that-- and I'm going to back up actually to this slide.

DevOps security?


DevOps security?
Development (Software Engineering)
Quality Assurance (QA)
DevOps
Technology Operations
Software Engineering Institute | Carnegie Mellon University
Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University
4

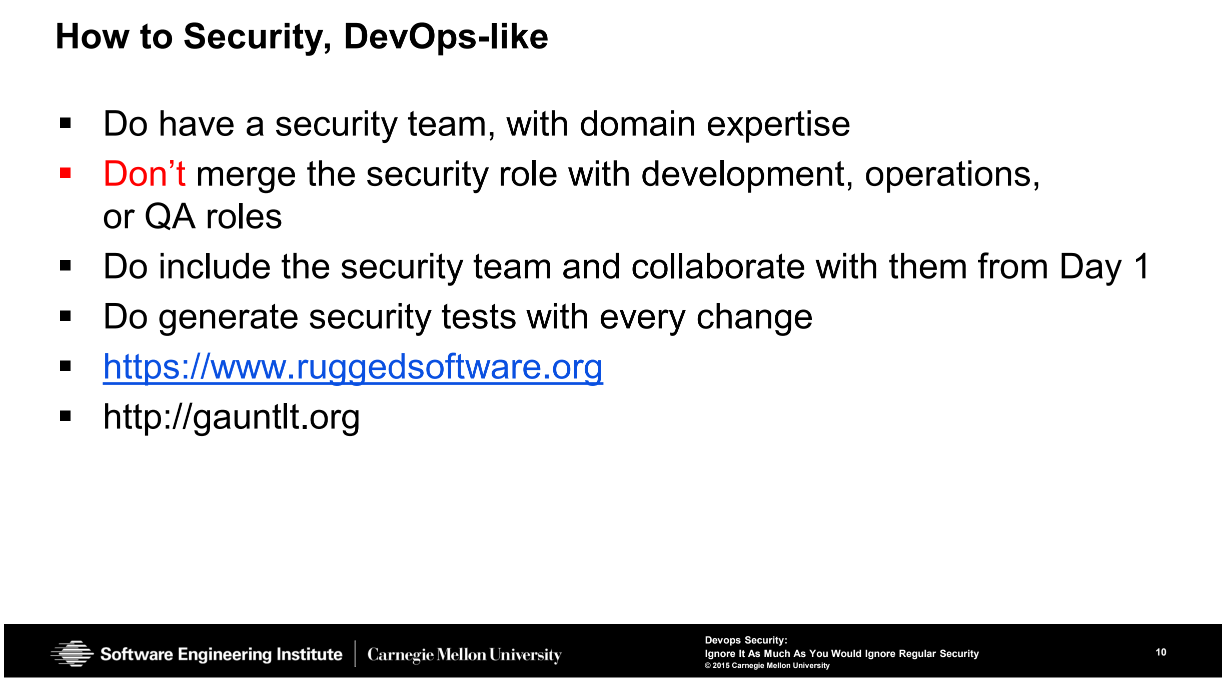
**004 Right, the Venn diagram. So, DevOps is not about merging roles. So, we're not taking out a particular role like development and saying well, you're also going to do release management, and you're also going to do provisioning of the servers. It's not saying this at all. The developer still maintains his responsibilities as a developer. The operations still maintains the responsibility as operations. And the same goes with QA. And the same goes with you have a security role defined in your organization or a security team, they're still doing security. That's their domain expertise. And so, that's what they should be doing. So, it's not about merging these into a single role or providing anyone with unilateral ability to just do whatever they want with the application. It's really about everyone maintains their separate responsibilities. But they're working more closely together.

Christopher: Right.

Shane: Okay. We got more questions. We'll come back to them throughout the presentations. We'll let you guys move on.

Timothy: Okay.

How to Security, DevOps -like



How to Security, DevOps-like

- Do have a security team, with domain expertise
- Don't merge the security role with development, operations, or QA roles
- Do include the security team and collaborate with them from Day 1
- Do generate security tests with every change
- https://www.ruggedsoftware.org
- http://gauntlt.org

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

10

**010 Okay, so how to security DevOps-like? So, this is kind of talking about security as you could address security with your project how you would normally. So, you have your security team, and they come in when they come in. And they're involved in the project. But then to do it in a DevOps-style, like we were saying before, it's all about shifting left. It's all about bringing areas of concern into focus from day one. And so, you're talking about the security person sitting at the desk from the first day of the project planning before any code is written, before the architecture is even thought about. That white boarding session when everyone's kind of brainstorming like how do we solve this problem, we have security people sitting in the room seeing that creative process going on.

Christopher: Right. That's pretty well thought of when you think about DevOps that we're going to add developers. We're going to add the operations guys to that mix. We're going to add QA testers to that mix. And this is just the idea of adding additional people. We're adding security guys to this mix.

Timothy: And it can-- like I say, there's concerns that it can be kind of hairy. There's a lot of people in the room. There's a lot of opinions. But if you have defined areas of responsibility, and this is more at the challenges of adopting DevOps, but if you focus on what you're supposed to focus on and everyone knows their role, and they know when to pitch in ideas, then they can work really well.

Christopher: To that point that there's a lot of people in the room, one of the big things you have to really think about at the very beginning there again is what are you building towards. What are your quality attributes? What does the business care about? What's acceptable? What's unacceptable risks?

Timothy: What sort of security do you actually want to attain?

Christopher: Right.

Timothy: How much effort do you want to put into that?

Christopher: Yeah, I mean some places are fine with their application being brought down by DOS occasionally. So, there's an acceptable level of those. It's just a DOS. It's not going to be worst thing in the world. Whereas some applications absolutely cannot be down. The longer they're down, the more money the company is losing.

Timothy: Right.

Christopher: And so, you have to think about that. Just those thoughts change a lot of okay, well we're going to build in this type of security versus this type of thing.

Timothy: And if the security folks are rolling into that conversation a month late, that's wasted time and effort that you put in toward implementing something that you may not need to implement. Whereas they knew what should have happened back then.

Christopher: Yeah.

Timothy: So, we have two links on the page. One is for rugged software, which is a concept. It's, as I understand it, very much a philosophy of developing software that kind of states that you go into software development understanding that code will break. And there will be people that try to attack your application if you're online, say. Or, your code will have bugs. And you will have problems with your code. And to design and develop the code in such a way that it is robust and it will withstand these problems that it's going to face. So, it's basically acknowledging the chaos that--

Christopher: Yeah, acknowledging that chaos and then trying to build into your application how are we going to be able to react to that. If there's a DOS, how can we get back up quickly? Or if we get popped in this way or that way, what is the system going to be able to do to react?

Timothy: Yeah.

Christopher: That's a lot of what that's about.

Timothy: Yeah, there's a lot of planning going into it, really a lot of understanding of what you're organization is about and how it's going to work.

Christopher: Right, which can be a little bit of a shift for some organizations. A lot of organizations, when they think about security, they think about different standards that they have to work towards. There's different things out there that prescribe we've got to go through this checklist, and then we're secure enough for what we're looking at. Sometimes that's true, and sometimes those checklists, they're not going to be built for your applications specifically. So, sometimes those checklists aren't sufficient.

Timothy: Right. And there's not a fix for everything, or a single fix for everything. It's very tailored.

Christopher: The other link that we have on there is to Gauntlt. Gauntlt is a testing platform. The idea is to create security testing as code that you can roll up whatever security testing you would normally have done into scripts that Gauntlt will run on a few tools. They support right now I think five or six tools including Curl, Nmap, and a few other ones as well as the command line. So, you can just write generic for the command line tool scripts that will run whatever tool just runs on the command line. And then you can plug that directly into your normal testing and integration process.

Timothy: Right. And you could automate that. You might not want to automate it completely. But you can automate it. Yeah, it definitely adds to the project testing.

Poll: Does your organization have a dedicated security role?

Poll: Does your organization have a dedicated security role?

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

11

**011 Shane: Okay, it's going to lead us to our next polling question. And I wanted to just address something really quick. We might have a blip with our connection. So, we've got a couple people chiming in that their video may be frozen. Just log in and out of the webinar with the same link you had before. And that will give you a new, fresh video feed. So, we are just acknowledging that a couple people are having that issue here this afternoon. So, the polling question we want to know is does your organization have a dedicated security role. And by that is it a CISO? Is that what you're looking for or what other roles may be dedicated security of people answering a question to know?

Christopher: Any role that has a direct responsibility to the software development process, that would be--- the domain expertise is the security of the application code itself.

Shane: Okay. And while we're giving you a couple of seconds to vote here folks, we're getting a lots of questions about if the slides are available to the presentation along with the archive. The slides are in the materials tab now. You can walk away with a PDF copy of the slides along with other SEI and CERT work on DevOps. And the event is being archived as well. The archive will be up most likely some point tomorrow, same login information that you used today. So, show our results here, and about seventy-two percent yes, they have a dedicated security role, fourteen percent no, and fourteen percent not sure. So, again I'm assuming you're looking for a higher number that-- a dedicated role.

Christopher: Yeah, that's good.

Timothy: Keep those security people involved.

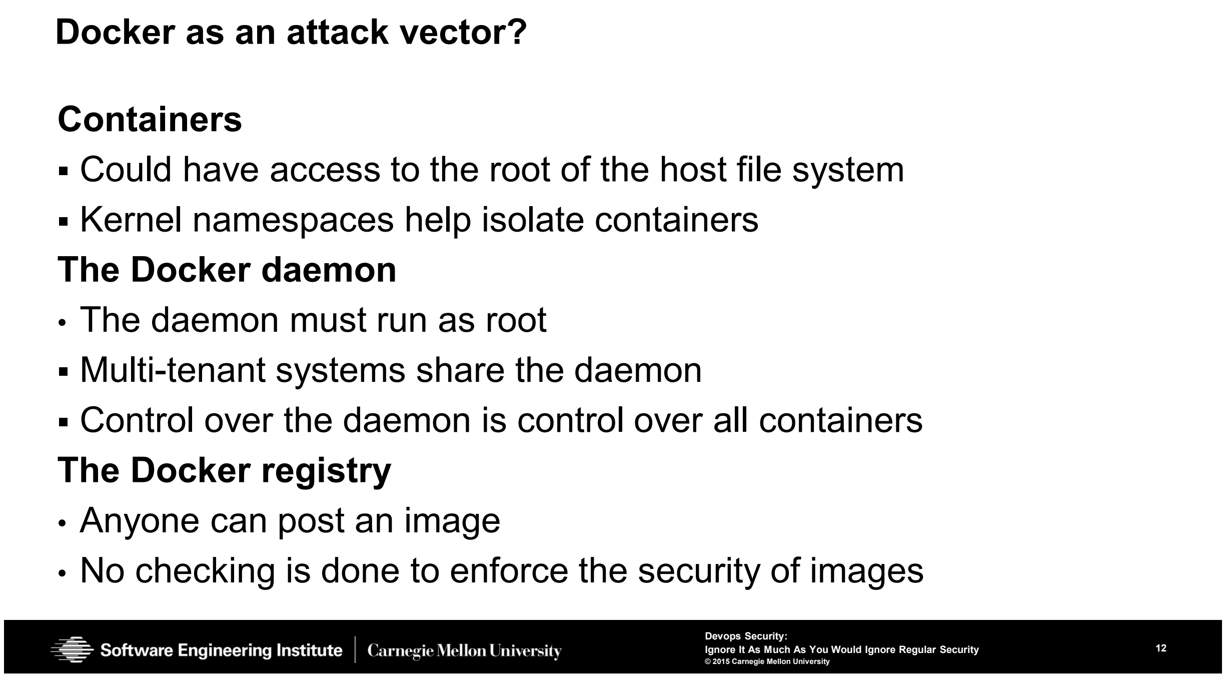
Shane: Yes.

Christopher: Yeah, and for those that don't know, find out. And start involving those security people as much as you can.

Timothy: Yeah.

Shane: Okay. Back to you guys.

Docker as an attack vector?



Docker as an attack vector?

Containers

- Could have access to the root of the host file system
- Kernel namespaces help isolate containers

The Docker daemon

- The daemon must run as root
- Multi-tenant systems share the daemon
- Control over the daemon is control over all containers

The Docker registry

- Anyone can post an image
- No checking is done to enforce the security of images

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

12

**012 Timothy: Okay, Docker, so Docker, a quick thirty second, if you're not familiar with Docker, if you haven't used Docker, Docker is a way of using-- it's basically using Linux kernel constructs to section off isolated execution environments for code. So, it's a way of quickly-- another way of saying it is a way to quickly deploy your application code as a running container to your customer or to release it to the wild. And a container basically holds everything that needs to run your software. And it's easy to package that container up.

And there's a Docker hub. It's a registry of the images basically. And so, you package your software up as an image. And you can push it to the registry. And other people can pull it down and run it without needing any other requirements on their system. Right now, I believe it is only Linux only. It doesn't run on Windows yet. But Linux is huge. So, it covers a lot of the bases.

But so, Docker is a very popular technology in DevOps. It's simply because of the fact that you can automate the creation of these containers. So, you have your application code. You can build a container that holds all of your application code. You can automate the deployment of that container. You can automate everything about it. And so, it fits in very nicely into the continuous integration, continuous deployment.

Christopher: And it's quick to bring them up and take them down. It's very portable, very fast. A lot people will compare them to virtual machines. And in some ways, that's true.

But in a lot of ways, it's not true. With these containers, you almost sort of get this feeling that you're running your code in this sandboxed environment, this completely isolated environment. And that's not always the case. It's not necessarily the case.

Docker is concerned about security. And they are trying their best to keep your processes from affecting anything else on that machine. And right now, they're doing this in a couple of different ways. But right now, the primary way they're doing it is with these Linux name spaces. And basically, Linux name spaces can allow you to-- your process to run in one of these Docker containers as if it's-- it'll really only see what's in that container, what's within its own name space.

Timothy: The name spaces are used to-- so, there's the aspects of running processes on a machine such as the process ID is managed by a name space. And so, to create a container, the Linux name space is used to provide an isolated path for all processes to be assigned a process ID within that container, so that they don't see other process IDs in other containers or on the host.

Christopher: And so this feels very much like a virtual machine. And at least for KVM, Linux's KVM, kernel based virtual machine that they have, that is a little different how that runs. That actually has hardware to help aid with that separation. It's not necessarily-- you're not just using the name spaces there. There's other stuff involved.

So, with Docker, what that means is-- what this name space means is that you're running those processes. You're still running on the Linux kernel. So, if your container, if something happens with your container and somebody takes advantage, runs a process that is going to subvert those Linux name spaces in the kernel, they could get access to your kernel. That's the potential-- that's something that could happen.

Timothy: And there was a way-- they had a release last June where there was a problem. If I'm remembering correctly, there was a hack where the-- within a container, a process running inside the container could access and modify things on the host inside the kernel and gain access to other containers, which was quickly patched. But things like that happen.

Christopher: Right, absolutely.

Timothy: And another way that you could use containers poorly, I guess, is you could share-- the idea is when you create a container, you share files-- you can share files from the host system that's running all of the containers into that container. So, you could share the root file system as, say, /host inside the container. And as root inside the container, you can modify any file. So, there's things that you need to be aware of if you're running a container registry, or an image registry. If you're running a Docker registry, anybody-- and you're allowing people to post images to that registry, and you're allowing people to spin up containers on that registry-- or spin up containers, there's nothing that says in Docker we're going to check to see what all these different containers are doing. That's really up to the people that are posting them.

And the same thing goes with pulling images down and running them on your own. You could pull somebody else's image down and run it as a container on your own machine. And who knows what it's going to do. You have to look at the configuration for the container and make sure that it's actually doing what it says it's going to do and it's not going to hurt your system.

Christopher: A little while ago, there was an article out saying that somebody had looked through the public registry, the Docker public registry, that the images in there, thirty percent of them were vulnerable. I don't how they came up with the thirty percent. But that's a big number. That's something to really consider when you're thinking about where am I going to get my image, am I going to pull it down from the public registry, or am I going to create my own.

Another thing that we really haven't mentioned yet is the Docker daemon itself. That would be the thing that controls all of these containers on the host system. And it has to run as root. There's no other option. So, you really have to be careful about who you're going to give access to this Docker daemon to. I mean you don't want to just let anybody have access to do things with this daemon without some vetting because they potentially could get into--

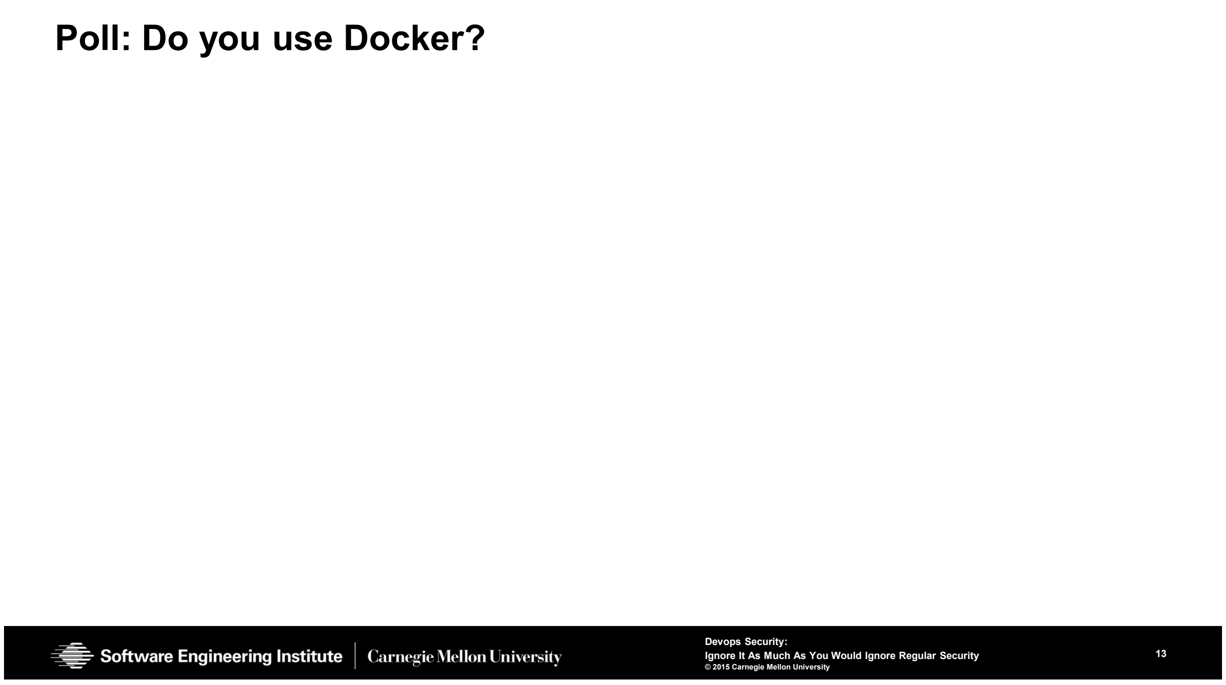
Timothy: Any container that's running on the host.

Christopher: Yeah.

Timothy: Docker recommends in their documentation if you want to run this on a separate host, that that host is dedicated to running the Docker daemon. And nothing else runs on that machine.

Christopher: Which, that's another thing that people running on a multitenant system out in the world should think about. Your container, you better trust the owner of that multitenant system because they're going to have access to your data most likely. Unless they're putting in their own private restrictions for themselves, they're going to have access to your data. And you have to assume they will.

Poll: Do you use Docker?



Poll: Do you use Docker?

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

13

**013 Shane: That's going to lead

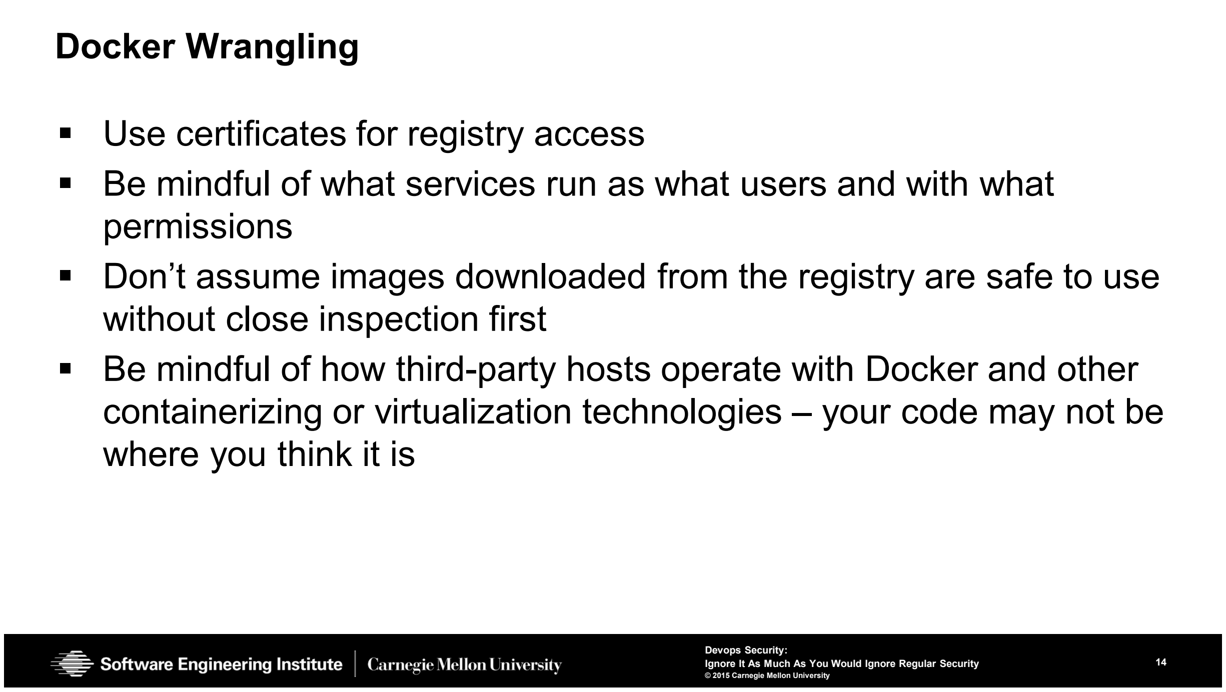
us to our fifth and final polling question just want to know-- we're going to pose it now is do you use Docker. So, we'll give you about fifteen or twenty seconds to vote on that. And while we're waiting for you to vote, I had mentioned a materials tab earlier. It's actually called a files tab. Beneath the video window, you'll see a files tab. That's where all the resources from today are located. Also a couple questions about if CEUs are available for logging in today. And they are. Just contact us after the event. We'll send up a follow up email tomorrow about where to find the archive and resources. And you can reply back to that about how to get CEU credit for today's attendance.

So, let's show our results on who's using Docker, eighteen percent yes, eighty-two percent no, not using Docker.

Timothy: Okay. I'd recommend giving it a shot.

Christopher: Yeah. It is a nice tool. With all the security concerns that it might have, it's still a very nice tool. And as long as you manage those security concerns and are mindful of them, you can use it very effectively and still be safe.

Docker Wrangling



Docker Wrangling

- Use certificates for registry access
- Be mindful of what services run as what users and with what permissions
- Don’t assume images downloaded from the registry are safe to use without close inspection first
- Be mindful of how third-party hosts operate with Docker and other containerizing or virtualization technologies – your code may not be where you think it is

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

14

**014 Timothy: Actually, we're going to continue talking about that a little bit and how you can use Docker. So, number one thing that you can do is if you're hosting your own Docker registry, you can implement a certificate between the Docker daemon and the Docker registry. So, that's saying that only the Docker daemon that you use-- and by the way, your Docker daemon could run on your development machine, your laptop that you run on. Or you could deploy it inside your environment. So, you have your own private daemon that can be shared.

And so, that Docker daemon needs that certificate to talk to the registry. And in doing this, you're kind of restricting how many people can actually talk to the registry, push images to it, pull images from it. So, by limiting your audience, you're kind of tightening your security a little bit.

Christopher: Right. Yeah, another thing you should be aware of is that Docker does have a security document that they put out that goes ever a lot of this information and a lot of the other information. Maybe we can get that link up at some point on there.

And I mean a couple other things that you should think about when you're with your Docker containers is regardless of where you got that particular container, you should be mindful of a few things. One of the things is you should know what you have running on it. What packages do you have running? What are the versions on it? Are they all patched and up to date? Are their security holes out in the world that have been found for any of your applications, any of your-- any of those packages.

Timothy: Any third party tool you pull in, you should always do some research on them to see what they're--

Christopher: Right. And once you create a container image, you really shouldn't be messing around with it for too much. Docker suggests that you-- that if you have to make changes, that you basically you create a new image, major changes to the image once you've finalized it. So, you can run-- Docker does have a diff command in which you can run that diff, and it will tell you this is what we have in the registry versus this is what you have running on your system. And here are the differences in there. So, that's a good tool.

There's also a read only option that you can run your container read only so that you know that nothing is going to get changed that way.

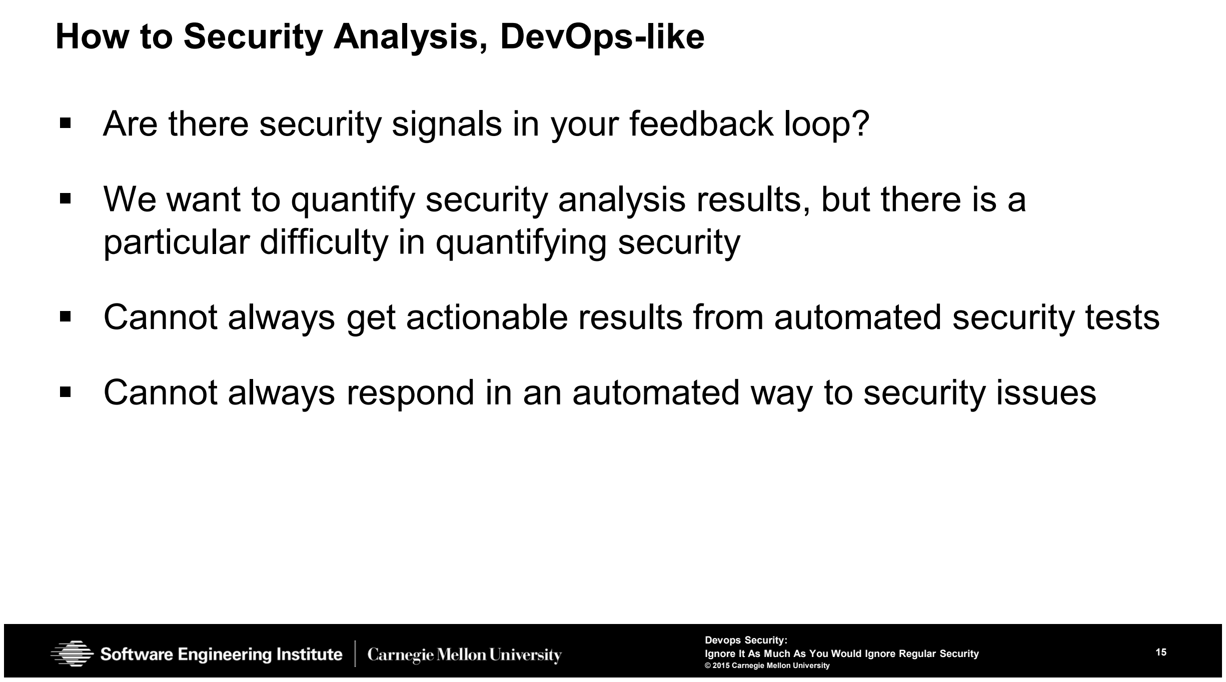
Timothy: By default, when a container is started, it gets a read/write layer. And so, yeah the option to say we don't want to write anything is important. Their documentation is actually pretty good. It's excellent. But of course, getting your hands to the keys is always the way to do it.

Anything else we want to talk about with Docker?

Christopher: I think we pretty much covered it.

Timothy: Okay, all right.

How to Security Analysis, DevOps -like



How to Security Analysis, DevOps-like

- Are there security signals in your feedback loop?
- We want to quantify security analysis results, but there is a particular difficulty in quantifying security
- Cannot always get actionable results from automated security tests
- Cannot always respond in an automated way to security issues

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

15

**015 So, security analysis in the DevOps space, so first of all, what are we talking about with security analysis? So, we're talking about looking at the application that we're building. And we're talking about how do we look at the software that we've built and determine if it's secure. You're basically making a determination of the quality attributes that you set out with initially to say did we meet all of the things we wanted to meet with our security team. Everyone had their input to it. And can we look at DevOps processes such as the measuring and the monitoring that are kind of prescribed with DevOps, and can we use those to determine anything about the security or application?

There's a lot of problems with that. And one of them is that by automating things, as you do with DevOps-- so you might automate security analysis testing. And you might automate running these tests to see how vulnerable your application might be. I'm probably not explaining this as well as you could. I'm going to give it a shot. But the results that you might get from this kind of security analysis in an automated way, looking for security vulnerabilities, aren't always something you can act on easily. It's not always obvious what you're going to do as a result of a failed test.

Christopher: Right, right. I mean it's very important with a lot of the continuous integration and a lot of the automated processes to have some feedback coming back at you. You don't want to just-- if it builds and if it runs, throw it out in the world or whatever. You want to be doing some tests on the new features that you've added, the new code that you've added every time that you're-- before or as this continuous integration is being done. And you want some feedback from those tests.

This includes security tests. And actually in a lot of ways especially security tests. I mean if you were running, say, a fuzzer to throw packets at your application and at your new features that you've just added, the information that's coming back from that fuzzer might show if I throw this packet at it, I get some information back. well, often, your security automation tool is not going to have any idea if that packet is going to give an intruder information into your system that they could then use to craft a malicious packet that would be able to give them access, full access, to your backend system let's say.

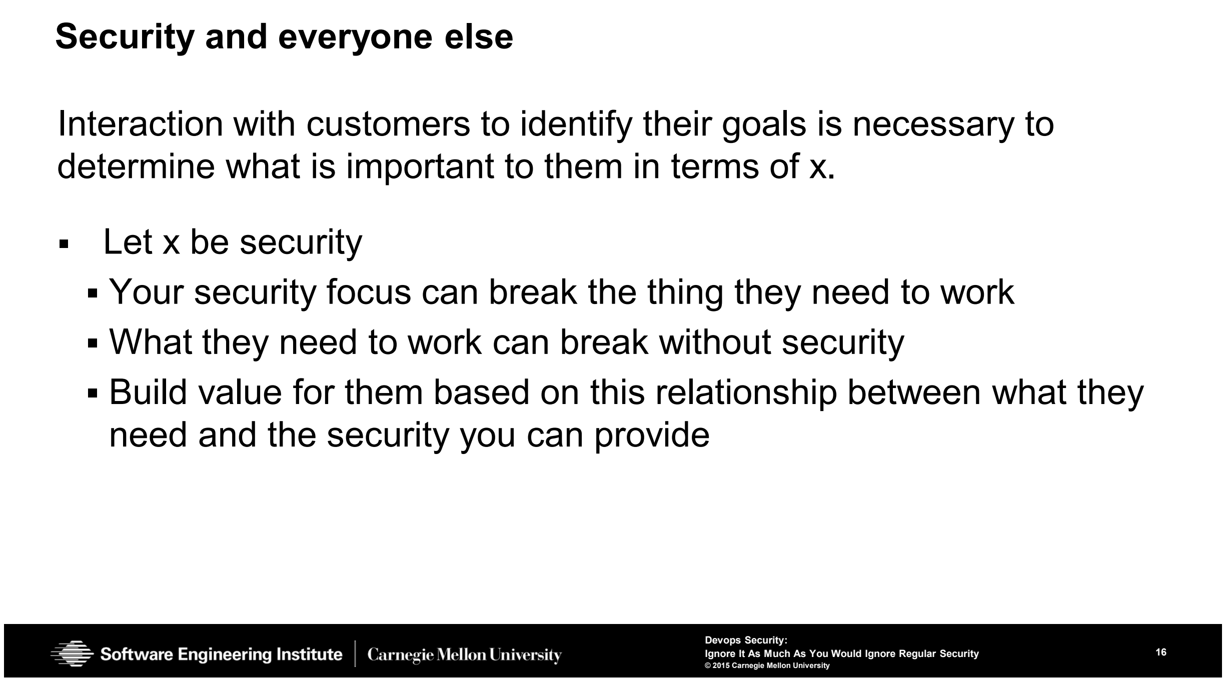
Timothy: It takes more of a human touch to look at these kinds of things.

Christopher: Yeah, it might just be giving, spitting back gobbeldy gook that honestly just doesn't mean much to an attacker. Or it could be giving back basically a way to get into the command line, a way to get root on your system. So, it's important that, as part of your feedback loop, that there's eyes on that, that somebody is taking a look. This was the results. This is why it passed. This is why it failed. And this is what happened, before just blindly assuming everything's either great or bad.

Timothy: And this goes back, again, to automate. Go forth and automate. Automate everything you can. But at the same time, always be wary of what you're automating. Pull back. Always have those gates. Human interaction with this stuff is really important.

Christopher: Yeah.

Security and everyone else



Security and everyone else

Interaction with customers to identify their goals is necessary to determine what is important to them in terms of x.

- Let x be security
  - Your security focus can break the thing they need to work
  - What they need to work can break without security
  - Build value for them based on this relationship between what they need and the security you can provide

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

16

**016 Timothy: Our last point is about everybody else that's maybe not directly on the project team. So, you have this idea. You have quality attributes about security that you aim for. And you have ideas about how you want-- how secure you want your application to be. And you, as a technical person, might feel very strongly about these kinds of things. But you also have people that are related to the project. So, you have business developers. And you have your customers. Or you have the general public. Well, you can do anything about the general public.

But you have a particular customer that you're shipping your product to. And you might feel a certain way about security, but that might not translate really to what the customer has as far as their requirements go for what the application needs to do. So, you need to build time into the schedule to make sure that your security is in shape. And they may not want to do that. So, you have to communicate with them in a way that conveys that importance.

Christopher: Basically, what we're saying here is push interacting your users or with your customers left. We're trying-- let's bring that interaction and try and do it upfront so that when you starting to design your system, that those concerns are forefront in your mind as you're designing and what you're building towards is actually going to be used.

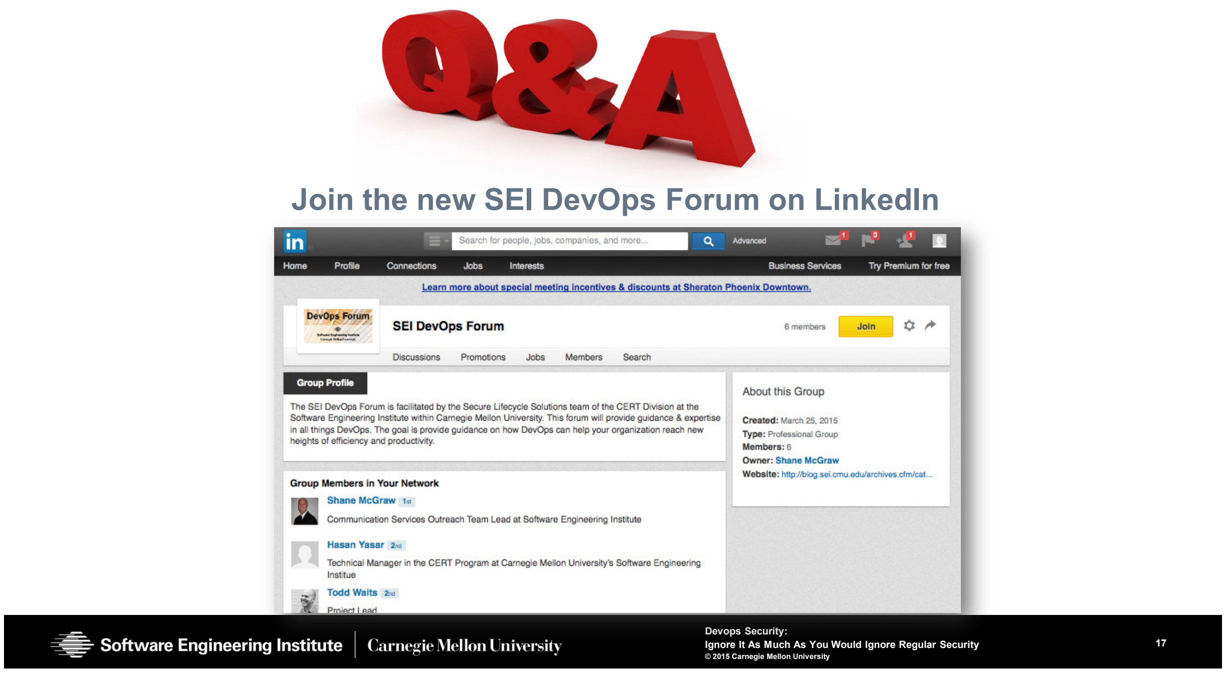
Timothy: Right, building the requirement for the system, what you're going to do, and if the customer is-- and the business developers and managers, if they're on board from day one, they'll see. It'll be transparent to them. It will be clear that you're aiming at security, that you're trying to get something done as far as security goes. And if they have a problem with it, I guess that's the time to bring it up. And you get that out of the way early, rather than face it later.

And security with interacting with customers can be, or people that aren't directly related to the project development itself, can be tricky because the thing that you want to do to make the application more secure might make it too restrictive or alter the user experience to a point where it's not what the customer wants anymore, which is why it's a good thing to get this out in the open early. And then by the same token, what they want, that feature that they want to push to their application might be a vulnerability. So, they might want to say let's put a text box on your web page that people can just type in whatever they want to, and the system has to do it. And that's--

Christopher: I mean it's good to communicate with the users and the customer and all that. You don't necessarily have to always go-- follow blindly whatever they want you do. at the same time, this is a really great time to make sure that the quality attributes that you're building towards actually make sense, and make sense for your product, make sense for your business, and make sense for the customer base that you're trying-- the demographic you're aiming at.

Timothy: Right.

Join the new SEI DevOps Forum on LinkedIn



Join the new SEI DevOps Forum on LinkedIn

Search for people, jobs, companies, and more... Advanced

Home Profile Connections Jobs Interests Business Services Try Premium for free

Learn more about special meeting incentives & discounts at Sheraton Phoenix Downtown.

SEI DevOps Forum

6 members Join

Discussions Promotions Jobs Members Search

Group Profile

The SEI DevOps Forum is facilitated by the Secure Lifecycle Solutions team of the CERT Division at the Software Engineering Institute within Carnegie Mellon University. This forum will provide guidance & expertise in all things DevOps. The goal is provide guidance on how DevOps can help your organization reach new heights of efficiency and productivity.

About this Group

Created: March 25, 2015
Type: Professional Group
Members: 6
Owner: Shane McGraw
Website: http://blog.sei.cmu.edu/archives.cfm/cat...

Group Members in Your Network

Shane McGraw 1st
Communication Services Outreach Team Lead at Software Engineering Institute

Hasan Yasar 2nd
Technical Manager in the CERT Program at Carnegie Mellon University's Software Engineering Institue

Todd Waits 2nd
Project Lead

Software Engineering Institute | Carnegie Mellon University

Devops Security:
Ignore It As Much As You Would Ignore Regular Security
© 2015 Carnegie Mellon University

17

**017 Shane: Okay, it looks like we're ready for some questions, I think.

Christopher: I think so.

Shane: Lots of good questions in the queue. Just before we dive into that, folks, I wanted to just take a second to invite everybody to, if you have a LinkedIn account, to join us to the DevOps forum. There's a LinkedIn group. So, if you are on LinkedIn and go to the search box and go to the group tab and just search DevOps forum. We invite you to join and continue the conversation from today or future questions you have about your DevOps related work. We have various staff monitoring that space. We'll be glad to interact with you there.

So, let's get into some questions. From Anon asking where does bi-model IT, if you're familiar with that or not-- I'll finish. Where does bi-modal IT sit in this conversation? Can the terms bi-modal IT and DevOps be used interchangeably? How are they related? So, are you familiar with bi-modal IT, first of all?

Timothy: Actually, I'm not.

Christopher: I'm not either, sorry.

Shane: So, Anon, maybe that's a question we log into the DevOps forum. And maybe there's going to be users out in the community that are. Or feel free to chime back in with some more detail what you mean by that term. And we'll repose it to Tim and Chris in a moment.

Let's go to Robert's question on how well does DevOps align with Agile concepts. Any important point we should be concerned about?

Timothy: Well, DevOps is actually a lot of times considered to be an extension of Agile. So, where Agile is focused on short sprints and being fluid and kind of ditching the waterfall approach to doing software development, we're-- it's basically doing that. But we're also including the operations folks. So, Agile doesn't strictly prescribe you need to talk to IT operations during the course of your sprints. But that's basically what DevOps is saying is we're going to basically doing Agile. But we're going to pull more people to the table. And we're going to shift left more concerns from all of these different groups.

Christopher: Yeah, the idea is that the tooling and the culture of DevOps sort of builds on Agile's culture. And together, they create a nice stack that will help the developers and everyone else involved create an application that's better.

Shane: Okay, Murray would like to know are the some development initiatives that are not suited for DevOps. Should organizations entertain using a multiplicity of development approaches depending on what individual circumstances dictate?

Timothy: I would say absolutely. I mean it's-- what kind of development project do you think DevOps would not work in? It actually doesn't-- I don't think it-- I think DevOps is a great idea for any project just because all it is is breaking down walls and collaborating more.

Christopher: Yeah.

Timothy: I mean that's one of the core features of it.

Christopher: I mean I could see certain aspects of DevOps, especially continuous deployment, which the idea is that you can press a button and your application will be deployed out in the world or wherever onto whatever deployment platform it would reside, I could see that not fitting in with some organizations. Some organizations might want to have their development environments air gapped from the rest of where they're deploying. That makes sense. But that doesn't mean that you can't do DevOps.

Timothy: Some aspects of DevOps, not all.

Christopher: DevOps really is much more of a cultural thing than a tooling thing. Some of the tools might not fit in. but I can't imagine a situation where the cultural-- where it's not a better idea to have your developers talking with operations people, talking with testers, talking with a lot of people at the beginning-

Timothy: The security folks.

Christopher: The security folks, yes absolutely, at the beginning of the project and throughout the life of the project rather than starting development, going for a little while, then talking to operations guys or security guys or whomever and bringing them in later. I can't foresee a situation where that's not a benefit.

Timothy: Yeah, I agree. But continuous deployment, maybe not.

Christopher: I mean, yeah, a lot of that stuff it's really dependent on what's your situation. You really have to take a good look at what am I going for, what's my environment, and what are my business needs, and does this fit in or not.

Timothy: And even-- so, we used the term waterfall earlier. And waterfall is associated with a software development process. We're gathering requirements. And then we're going to design the system. But waterfall in the way we were using it is more of like just the overall software project. So, we're going to do all of the development. And then we're going to do all the testing. And then we're going to do all the...and that is-- the waterfall method for software development, where you do the requirements gathering. It's a sequential process. It's very linear. And that does apply to some projects.

So, Agile is not for everybody. The multifaceted kind of structure for developing a project I think works. And you could apply DevOps to waterfall. There's no reason that you couldn't do a sequential project where you do all your requirements gathering upfront, but everyone's sitting at the table, right? Everyone's talking about it. And then you go into the design phase. And so, there's no reason you couldn't match that together with anything, and DevOps will work that, too.

Christopher: Yeah. I mean I think DevOps really was designed to go-- to work very well with Agile. But absolutely it can work with just about anything.

Timothy: The cultural aspects of it.

Christopher: Yes, absolutely.

Shane: Okay, great answer. Another question here, how should conventional security roles of application and infrastructure evolve to adapt the DevOps paradigm? Do you see Sec DevOps emerge as a new concept?

Timothy: Yeah. That's a term, DevOps Sec, Sec DevOps. But yeah, that's a term that I've heard before, that we've heard before. And so, yeah absolutely.

Shane: Okay. Next question from Newper asking Amazon pushes a change through their continuous delivery pipeline every eleven seconds. Could you discuss how you could do security in that pipeline?

Timothy: The way we're talking about it, having that human interaction we think is very important. So, eleven seconds--

Christopher: You might not be able to do it in that pipeline itself. I mean in the continuous delivery pipeline in that automation. But you definitely want to do it before you pushed. You'd want to make sure that when you're building-- I mean anytime that you're building a new feature into anything or changing a bit of code, you want to make sure that you're testing what you're changing. You want to write unit tests to make sure that what you've just done actually fixes what you think was broken, or provides the feature that you think it provides without any issues. Whether you do it inside of an automated continuous deployment, inside of that particular infrastructure, or whether you do it before that, doesn't necessarily matter. But I mean certainly there should be some tests done to make sure that you're not introducing any security flaws. And you're not introducing any other types of vulnerabilities into that system.

Shane: So novice question from me, and maybe this is what you were just explaining, so if you're bringing someone else's code in, is there a way to make sure it's secure because I know we have secure coding standards here. Is there a way to apply those to someone else's code to make sure it is secure? Or is that what it does?

Timothy: I mean, yeah, there are ways. We were talking a little bit earlier about Gauntlet and a lot of the security. A lot of the testing tools that it uses, you could use something like that to take a look at your entire application. There's that stuff.

There's also, like you said, we have the secure coding initiatives. You could use those tools. There's also fuzz testing that can be done on your application. And a lot of that obviously would happen in integration rather than deployment, at that time.

Shane: Okay, Liz would like to know, they have a separate security team-- it just says have a separate security team. Would you not have security functions working continually within the delivery team rather than siloed? Would this not lead to better buy in from the delivery team where some security as a drag on their velocity?

Timothy: If I'm understanding that, the security analysis would slow down the pace of the release cycle.

Shane: That's how I'm reading it, yeah.

Timothy: And that's-- this kind of relates back to the Amazon question, previously, this short release cycle. And just because-- this is the way I'm thinking about it is a release every eleven seconds, first of all, I don't know that that necessarily means that there's a change in the code every eleven seconds. They might be just deploying the latest most tested stable release version of it. But that also means that developers committing code at the beginning of, say, a pipeline, there's-- I doubt that the developers commit code. And then eleven seconds later, it's deployed. It's probably committed. And then security can happen within that space. And there might be a delay while the security happens. But it's built in to the process.

And so, as long as that's accounted for and that's built in, it really shouldn't get in the way of operations or the release cycles because the release cycle, you could think of it as the releasing of the code is pulling from a different bucket. It's not pulling from freshly committed code.

It's pulling from another bucket that's further on down the line that has had eyes on it that has the security testing standards applied to it to make sure that it's stable.

Christopher: There's also a little bit of a perception that security testing takes a long time, which can be true but isn't necessarily. If you're really mindful about what type of security that you're concerned about and what your application currently looks like, when you're changing the code and writing those new security tests, you can be very specific about what you're testing and how you're testing.

Things that might start adding a lot more time to that might be things like fuzz testing. But again, you can start-- you can pinpoint and really sort of tailor that fuzz testing towards what you want to exercise. What did you just create? What is important to you right now? Rather than just saying I'm going to leave this fuzz tester go on random for all night before I can deploy my code.

Another thing, too, is that some places might find that it's more important for them to get a fix out right now out the door right now. And if that fix leaves them vulnerable, that's acceptable risk for them right now. I mean the way that this stuff works and how automated it is and how quick it is to deploy that might be fine. So, you can push out your important fix, and then in the background can start doing testing and things like that. You'd have to make sure that whatever your business requirements are and whatever your acceptable risk is that that falls within that before even considering doing something like that. But I mean that might be a possibility based on their scenario.

Shane: So, I know we're at 2:30, but if we can work in one more question as quick as we can. So, from Debenyu asking what is the best way to bring about DevOps transformation in a large traditional company that uses COTS vendor apps and presentation tier with SOA with legacy mainframe based backends. Change percolates through each tier using different generations of technology and teams.

Timothy: Okay it sounds--

Shane: We have a quick answer for that.

Timothy: It sounds like a bit of a mess. But it's a challenge worth addressing. Culturally, the stack is fairly irrelevant. Culturally, you can bring about a lot of good DevOps change simply by restructuring the way that you hold meetings, restructuring the way that you communicate, restructuring the way that you plan features for release. There's a lot of change that can happen without addressing the technology stack.

Christopher: You need buy in. you need a lot of buy in. I mean you need to get everybody on board, top down. That's really important.

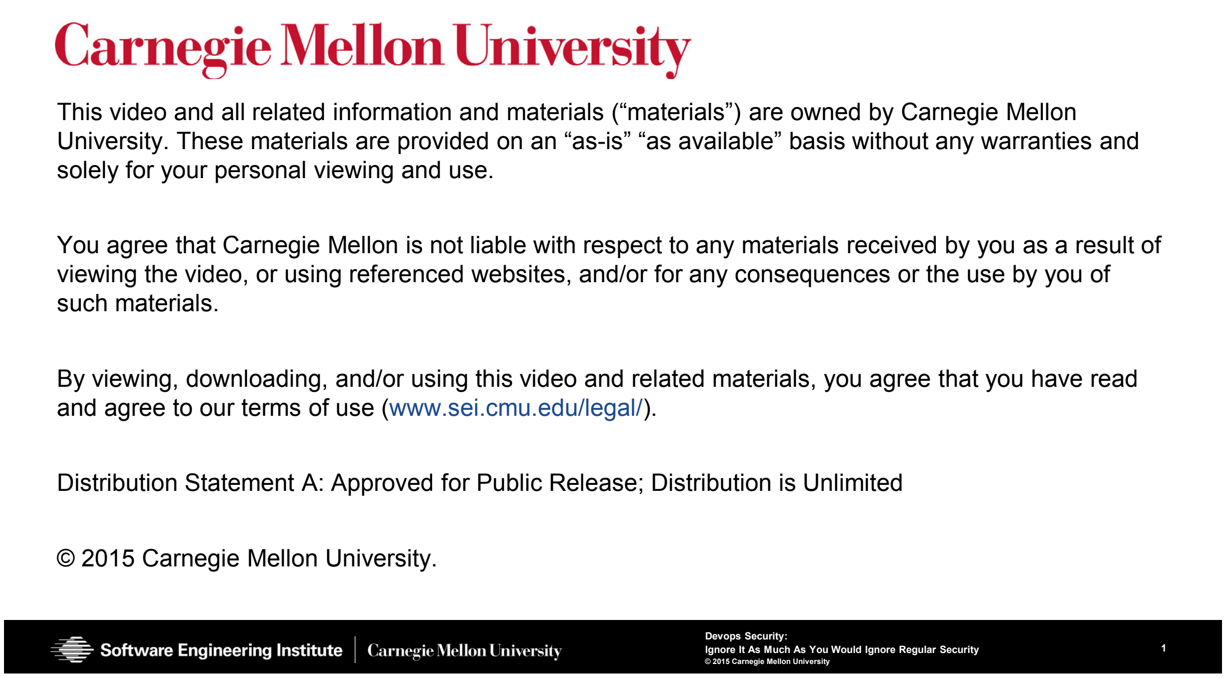
Timothy: That's the challenge. That's the challenge. And it's really dependent on-- it's a personality thing really. I mean there's no silver bullet to fix a problem like that.

Christopher: Yeah, that's a hard one. Good luck with that.

Shane: Tim, Chris, thank you, excellent presentation. Thank you very much for your time. Folks, that's going to wrap up our presentation today. And thank you again for attending. Upon exiting, we ask that you fill out that survey as your feedback is always greatly appreciated. And just a last reminder, if we didn't get to your questions today, please go on LinkedIn, search for that DevOps forum and post your questions there. And we will continue the conversation. Have a great day everyone.

Excellent job, excellent, excellent. You guys are excellent.

Carnegie Mellon University



Copyright 2015 Carnegie Mellon University