**SEI Webcast**

***What is Cybersecurity Engineering, and Why Do I Need It?***

**by Carol Woody and Rita Creel** 

**Shane McGraw:** Hello, and welcome to today's SEI webcast, What is Cybersecurity Engineering, and Why Do I Need It? My name is Shane McGraw, outreach team lead here at the Software Engineering Institute, and I'd like to thank you for attending. We want to make our discussion as interactive as possible today, so we will address questions throughout today's talk, and you can submit those questions in the YouTube chat area and we will get to as many as we can.

Our featured speakers for today are Carol Woody and Rita Creel. Dr. Carol Woody is a principal researcher for the CERT division at the SEI within Carnegie Mellon University, and her research focuses on building capabilities and competencies for measuring, managing, and sustaining cybersecurity for highly complex network systems and systems of systems, and she has successfully implemented technology solutions for CERT's diverse domains as defense, government, banking, mining, manufacturing, and finance.

Rita Creel is the acting deputy director for the CERT division of the SEI at CMU, and she has over 25 years' experience in software-intensive systems engineering and acquisition, cybersecurity systems and software measurement and analysis.

Now I'd like to turn it over to Dr. Carol Woody. Carol, good afternoon. All yours.

**Carol Woody:** Thank you. Thank you, Shane, and glad you folks could join us today. Rita and I certainly think cybersecurity is an important topic. Much of what gets publicized really points to instances of cybersecurity failure. What's less talked about but really more important is the engineering that needs to go into the systems and software to begin with so that the attacks aren't successful. We think of this as cybersecurity engineering.

Today we're going to focus on the key aspects of cybersecurity engineering and show you why this is important. Then we'll discuss ways in which additional rigor from the engineering perspective can reduce the potential of cybersecurity failures.

Engineers envision, design, and build the technology we rely on. A part of this effort is considering how the technology will accommodate cybersecurity. This consideration is needed throughout the lifecycle, from requirements through development into sustainment. This requires applying the rigor of engineering in preparing the technology to handle the operational environment where it will reside. Rita, I know this topic's near and dear to your heart. I'm sure you want to add to this.

**Rita Creel:** Sure, sure. Thank you, Carol, and welcome everyone. As the Venn diagram shows, cybersecurity engineering is a discipline that's at the intersection of system and software engineering and operational security. So system and software engineering encompasses the activities, the resources, processes, and information that are involved in turning the idea of some

capability into a system, an application, a service or a product that delivers the capability. So that includes the requirements, architecture, design, implementation, verification and validation and deployment and operations and maintenance, all those activities.

Operational security includes the activities, products, processes, resources and information that are required to protect and defend a capability once it's in operations. So it includes not only protecting the capability but also protecting and defending the mission or the business process that the capability was designed for. So in operational security, we're concerned with enterprise risk and resilience management, and identifying the attackers and threats and mitigating vulnerabilities, and in responding to incidents and attacks.

Cybersecurity engineering applies risk analysis and all of the things that are informed by operational security to the engineering activities and across the entire lifecycle to reduce the risk to the operational mission from cyberattacks and incidents.

**Carol Woody:** Through our efforts working with a wide range of government programs, we have identified six essential activities where cybersecurity engineering is needed. Engineering performs many of these activities, but cybersecurity engineering brings a specific perspective to the activity that is critical for today's operational systems. Let's look at each one of these individually.

Appropriate consideration of cybersecurity risk, especially early in the acquisition and development lifecycle, is paramount. From what we see, few programs consider more than cost and schedule risks. Few engineers understand the ways in which their designs enable attackers to bypass many of the standard security controls. One of the key ingredients in risk consideration is software. As a primary system component, it brings unique risks into the system based on how it is built and used.

Engineers design how components and systems will interact. Well-designed interactions should monitor the information flowing among the components to ensure well-formed expected data will be accepted, and ill-formed data or unexpected data will not be accepted. Cybersecurity engineering should be on the lookout for gaps in terms of designs and structure and requirements to ensure that unexpected and unwanted interactions among components and external systems do not occur. Rita, do you have something to add on that? I know interactions are something we talked about a lot.

**Rita Creel:** Yes, Carol. One of the things we find when we're looking at these vulnerabilities that come in every single day is that a large portion of them are due to the interactions between different components in the system or in a technology. So that's one area that we look at very carefully when we engineer a system-- all of the different components that the system interfaces

with to deliver its function. And I think that leads us to the next topic, and that's evaluating trusted dependencies.

So when you're engineering a system, you have a lot of resources and components that you're using and which you depend on. Some of these you don't trust and so you pay a lot of attention to them. But then there's a set of what we call trusted dependencies, such as accredited systems and cloud services, libraries that you may have in house that you tend to run tests on all the time, but you need to pay attention to these trusted dependencies as well. For example, they change over time. There's changes to what cloud services provide. You don't exactly know what's going on in a supply chain for a third-party service. So you need to have an approach to evaluate what's happening, whether it's through monitoring or periodic reviews to make sure that you can still consider this a trusted resource or component.

Another thing to look at is whenever you're using something shared, that's used by a number of different entities, even if it's, "Oh, we trust this vendor. It's a big name. They're really good in security," you have to recognize that there's a potential for many different individuals to touch that resource, so you need to pay attention to that when you're assembling resources in your engineering processes. Carol?

**Carol Woody:** Let's look at the--

**Rita Creel:** That brings us to the next point, which has to do with anticipating attacks, and that's something that's important in the engineering process as well. There's a concept of a blue team that thinks like a red team. As you're going through every step of the engineering lifecycle, you need to consider what could and attacker do. If I have a set of requirements, a set of capabilities, where is the possibility for an attacker to intervene and compromise my mission. So that's one very important area for us to look at.

Another thing to note is we were just talking about trusted dependencies and shared services. Attackers are maybe motivated to get the biggest bang for their buck in some cases. If you're a large company of if you're a government agency, they're going to look for places to attack where one attack can touch many, many different organizational units. So again, we look at an app store or cloud services or things of that nature. So, again, that all factors into the engineering lifecycle.

**Carol Woody:** We like to think that engineering focuses on how a system should function, but cybersecurity engineering needs to focus on how the functioning system should respond to an attack. That's another piece that needs to be thought about and planned for, as well as relationships with potential sources of attack.

Another aspect that's critical is the good engineering throughout the lifecycle. We see lots of point solutions that provide partial results. Good engineering starts with a good design, and that's carried through the building and verification of the system into its testing and fielding. Cybersecurity engineering should augment the design to consider what could go wrong and monitor the building and verification to confirm that the system is prepared for potential misuse and abuse. This is an area that we see frequently is overlooked in the rush to get things out and field systems effectively, to the detriment. We learn on the operational side that something is missing.

Slide 12 is one that we want to really emphasize, and that is measuring your cybersecurity. Engineering measures how a system performs and how the functionality is executed in terms of speed and reliability and a lot of the other aspects. Cybersecurity engineering needs to measure the unaddressed risk to ensure that it remains within tolerable bounds when the system is implemented. This would include issues with quality that could lead to unstable execution and limits in handling known threats, which could lead to opportunities, again, for the attacker to take advantage of.

**Rita Creel:** One way to factor in measures into your risk assessment is to just look at what the risks you have are and try to frame questions around those risks, about, "How could this particular function be attacked?" and then start to drill down and develop measures that will let you track that throughout the lifecycle as you're in your engineering practices.

**Carol Woody:** One of the questions that keeps coming back to us when we've been working in this area is, "What should we measure?", and one of my takes on it is that you're already measuring a lot of things. Look at what those pieces can tell you about your cybersecurity. This is a way that putting that spotlight on cybersecurity can really give you leverage in terms of better understanding what you are already doing so that it's not necessarily an additional activity, but it is focusing and having someone responsible for doing it that we see as a critical aspect. Before we continue, Shane, is there anything coming in that we need to worry about in terms of questions yet?

**Shane McGraw:** No. Just to remind everybody, feel free to get your questions in, but nothing in the queue yet, Carol.

**Carol Woody:** Okay, great. Let's next then consider the way we build and field systems that drive the critical need for these six cybersecurity engineering activities that we just described.

In every step of the acquisition and development lifecycle, we potentially introduce weaknesses that can impact mission execution. Engineering focuses on the creation of the final product, but cyber engineering needs to be there to consider how the system could fail and the ways that engineering can be applied to reduce that risk when we actually field the system. That requires

us thinking about not how we build the system but what could break it, and that's a different perspective that is not always easy for engineers to flip back and forth. That's why we've been emphasizing that someone needs to be in charge of this perspective.

One of the major contributors, as we've mentioned earlier, to design and coding issues is software, and a key aspect of that that also adds to the challenges is the extensive reuse. We have legacy code; we have third-party code; we have libraries; we have services and applications that are all tied together; we're integrating in open source, in some cases; and each one of these could contribute to a possible failure.

But it's not as dire as it sounds. What we've got to be looking at is what is the level of trust we're putting in each one of these sources. How are we tying them together? What are the pieces that integrate them? And when we integrate each one of these elements into the system, we're moving further and further away from a built-for-purpose structure, which is what we frequently think of when we design. So we think, we design it, then we go out and build it, but in reality we're bringing in all these other additional pieces, linking them together, and what we're ending up with can have unwanted functionality and also unknown defects.

This may improve cost and schedule, but we have to recognize it comes with a risk, and then we have to start measuring that risk and understanding how much is tolerable so that we can keep our end result within tolerable bounds.

Third-party components are now widely used and we have an example here for you to get a sense of the scale that we're talking about. In this automotive example, these engine control units are pre-structured components that are inserted based on the design, but the level of engineering rigor of each one of these components is outside of our available knowledge. They also include a million lines of code, unknown quality, and these are all inserted into the system as we build it, design it, and then ultimately field it. Each one of these particular components can also include other third-party components, extending the supply chain into other organizations of unknown source. So this is a major complex area and a potential entree for attack that we need to be concerned about. Cybersecurity engineering needs to consider the level of trust and the risk that these components represent, and how the design should accommodate them if we are concerned about that risk.

**Rita Creel:** Carol, we have a question if you're ready to take one.

**Carol Woody:** Sure.

**Rita Creel:** Question from Leon: Do you see cybersecurity engineering as a standalone function-- for example, in the DevSecOps stack-- or is the concept embedded in the various functions making up the DevSecOps stack?

**Carol Woody:** Well, I think it's a combination, because it needs to be embedded as part of the pieces and the flow as you're building the product, but particularly for the DevSecOps environment, you have to be looking at how you're building your pipeline. What is your pipeline protecting you against, and have you considered the right level of threat through your pipeline that you're actually focused on? You also have to remember that the pipeline represents a system unto itself that has to be maintained. So if you're integrating tools, those tools have to be maintained. It's not like you build it once. All of this has to be supported and maintained over time, which is why we emphasize the cybersecurity so there's a set of eyes that's really worried about that, and how does that integrate with all the other pieces. Because you don't want them to be magically updating all of your analysis tools when you're in the midst of a major build, and you want to make sure you've got the consistent product coming out. There's a dance of integration here that always needs to be considered.

Looks like we have a second question too, from Patrick: Should cyber engineers have ethical hacking skills, and if so, what other skills should they have? That's an interesting one.

**Rita Creel:** Yes. Well, I think-- one of the things that was mentioned earlier is the concept of a blue team that thinks like a red team. So you're building a system, you're hardening it, you're trying to protect it, but then as you're doing that, you should have a concept of, "How can this be attacked?" So if the cybersecurity engineers themselves don't have those skills, they should leverage the skills of ethical hackers to really put the system through its paces; and I think also that the hacker mindset helps in the entire lifecycle process, looking at the requirements-- looking at the requirements, not just the security requirements, but the requirements for the system and for the capability. I'm thinking about if I were an attacker, what capability would I want to disrupt and how might I do that, and how are these capabilities put together, and then start to build a program that really looks at the operational mission scenarios or the business processes and thinks about what do I have to do to test the security.

**Carol Woody:** Adding onto that, I think there's one other aspect that it's useful for cyber engineers to understand, and that is how do these attacks occur in the operational environment. We've looked at a lot of attacks to understand how they're getting in, to leverage that knowledge so that we can take that back in to better build the system the next time. That is one area that we see is not being used as effectively, which becomes the feedback of, "What does the operational tell you that you can then take in to build better the next time?" Some of this comes from the separation we have of people that build and then people that operate, but in reality, it's one integrated environment, and they need to learn from each other and share information. So I think it's in some cases not so much training as just understanding and having been exposed to the issues. Let's see where I was.

**Rita Creel:** Anyone can write software.

**Carol Woody:** Yes, that's true. We're teaching the children.

**Rita Creel:** Right.

**Carol Woody:** And what we're also dealing with is that we have a software industry that has grown so quickly over the years, over the recent years, that software is being produced faster and cheaper, but by uncertain skills in terms of the knowledge of how to do it is securely. That is not prevalent. It's not part of the teaching environment; it's not part of the learning that you'll see in these websites. So we know that engineering is critical to how these pieces are put together, but the engineers have to be mindful of the limitations that they have in the software and how do they either figure out how to remove them or work around them. That's part of the design process.

**Rita Creel:** I would like a How to Teach Your Kids to Code Securely class.

**Carol Woody:** That would be a good one. I think a lot of folks are hoping that the tools will take care of the problem, but if you really don't know how to use the tools and you aren't trained on the tools, there's limits to results that you're going to get out of it. There's a lot of knowledge that a developer needs, and we have to recognize that software is everywhere; no matter what you touch, you've got it. And the other piece that we have to remember is that no matter how good the developer is, the software has defects, because this is much more of a craft than it is a robotic, instrumented process that produces perfection.

We've got some analysis that was done by Capers Jones a while back, but he published numbers that are useful for us to think about as a became scale in terms of the fact that even the best of code has certain defects, and our research has shown that about 5 percent of those defects are actually vulnerabilities. So within every piece of code there is some level of vulnerability.

**Rita Creel:** A question, Carol, that's related to that-- it's a question from Carlo: How do we regard the risk of the current software estate-- so all the legacy stuff that we have and stuff in production-- versus recent engineered systems based on modern processes, including cybersecurity engineering? So what do we do with all the legacy?

**Carol Woody:** Well, I think it raises a serious question of how much legacy you should carry forward in your new systems. That's one of the pushbacks I give to the design teams when I'm working with new programs, because they waltz in saying, "Well, we can have 50 or 60 percent of this that we can adopt from our previous system." Well, that's great if it fits the security requirements you want, if it ties to the design, but if you're integrating it with a whole new set of new technology and you're just assuming that these pieces are going to work together or you can create some sort of integration that ends up being more of a Rube Goldberg interchange, then

what you're creating are ways for the attackers to move in on you if you're not careful, and few engineers really think of it in that perspective. They think of, "Oh, that's inherited risk." "Oh, we know how that software works. It works just fine." Well, a lot of mistakes have been made by assuming that you knew how the old software because you understood the old environment, not the new one. So it's something that you have to be very careful about.

When you think of the attacker, think of it in terms of three things that they need. They need a vulnerability, and we've already said we know there are lots of those with millions of lines of code, and a chunk of those are known vulnerabilities. They need access, and as we integrate, interface, add trusted connections, tie things together to shared services, all of those pieces escalate the access capability. And then they need to have the tools to exploit that, and as we increase our development tools and improve the way we build systems, those same tools are useful to the attacker, and as more and more of our code is publicly available through third-party and online, the attackers actually may have more time to study it and find the defects than we do, because we're trying to get another job done. So that puts us in some cases at a disadvantage.

**Rita Creel:** We have a couple of questions, but I think that we're going to come back to those in the next section. We'll be dealing with those.

**Carol Woody:** We'll cover those. I wanted to show this one slide to give you a sense of the scale of these issues. If you pick up any paper and start looking you'll see at some point somebody's talking about exploits and attacks, but these examples should be noted that attackers are successfully exploiting vulnerability in operating systems in applications, they're extracting data from high-value sources, and they're impacting the ways critical capability is delivered, such as healthcare services and utilities and things like that. All of these are now becoming vulnerable because of the integration of legacy, the lack of attention to cybersecurity, and the lack of rigorous engineering in the systems that are fielded.

You may laugh at this picture, but I'm afraid it's a little too close to reality for those of us that have worked on the operational side, in that it's very much of a reaction environment. You have to assume that you have been attacked, you are being attacked, you will be attacked, and act accordingly, but operational tools are not going to cure everything. This is just a stopgap measure. What we really need-- and this gets back to an earlier of what's it going to take dealing with all this legacy-- what we really need is good cybersecurity engineering to reduce the level of attacker success, and that's something that is just going to take discipline for us as we're going through these programs. Did I miss anything, Rita?

**Rita Creel:** No. Well, there's lots more to talk about, but--

**Carol Woody:** So far, so good.

**Rita Creel:** Just thinking about legacy some more, back to the dependencies in component interaction, that's where a lot of times the use of legacy can be problematic, because you have these assumptions means you don't dare touch the legacy, you kind of know what comes out of it and you know what inputs you have to send to the legacy. No one dares touch the code itself, but you're putting it in a different environment and the behavior is uncertain. So you have a lot of times errors that can lead to a successful attack and a compromise.

**Carol Woody:** Yeah. What I'd like to do next, and I think it's going to address some of these questions that I was looking at in the chat, is really focus the remaining of our time on how cybersecurity engineering can reduce the challenges we have for our critical missions.

We have assembled six ways that we feel like cybersecurity engineering can strengthen operational development; essentially, going back to some of the earlier things we mentioned: enhancing design, improving requirements, focusing on risk analysis, introducing measurement and lifecycle integration for the key aspects of engineering so that we're really bringing that cybersecurity perspective into the design rigor, so that it's considered one of the elements of the tradeoff discussions that are constantly occurring among different aspects that we want in the system.

Let's look at each one of these in detail so you can better understand why we think they're important, and hopefully you'll agree with us.

The design aspects of cybersecurity are critical. Everything starts with a good design. We are seeing that design weaknesses are a major distributor to software vulnerabilities. While they only represent about half of the reported software vulnerabilities, when you look at the most dangerous ones that have been defined, they are the vast majority of these, and without effective cybersecurity engineering early in the lifecycle, these design weaknesses persist. They can't effectively be patched; you have to essentially redesign, and the typical emphasis on cost and schedule means that redesigns are typically avoided, and so these get into the operational environment at a very high level and become much more difficult to deal with. This is where a lot of our reactive issues come into play, and you can only do stopgaps against them. Ultimately you have to go back and rework them.

We don't see when somebody's building on legacy that they're actually looking at the operational environment to see if there are design weaknesses relative to that legacy they're going to adopt again, and so we're perpetuating a lot of these problems by carrying them over and not really looking at the details of the design and understanding what the risk aspects are.

And some of this, I think, relates to gaps in security requirements. I think, Rita, you were emphasizing that one when we were talking.

**Rita Creel:** Yes, that's right, because your design is going to come out of your requirements for your system, your capability or service, and so security requirements are an essential foundation to have a secure product at the end of the lifecycle. So one of the challenges that we see is that for such a long time security was information security and it was separated from the system and software engineering process, and we still have a lot of that. So we have security controls that come in as requirements, but they aren't sort of in lockstep with the actual engineering process of figuring out requirements and the requirements trades, and they may be a set of security requirements that, "Yeah, we have to have these because we're mandated to have these," but they may not relate to the risks of your operational environment, the expectations of your user, and the way that you're designing and building your system. So you tend to have a lot of gaps in the security requirements. We see that a lot, and of course that flows down to design. So you have the design weaknesses that Carol talked about, and those flow into code and then we're playing whack-a-mole with the defects and whack-a-mole with the attacks throughout the lifecycle.

So what we want to do instead is to look at the actual risk. Yes, we want to take on the security controls that are required of us, but then we want to also look at the operational environment, at the mission, and at our system and our product and how it's designed and assembled and the different components, and trace through and see where could we be attacked and are we vulnerable there, and what can we do, what can we include in our security requirements, so that we can design a more robust system and have a more robust result, and that's something that Carol is going to talk us through next.

**Carol Woody:** Well, before I go, I wanted to say something about a couple of questions that have come through. Brent K, for example, mentions a concern about hardware and how they deal with the requirements. We don't view hardware so much as isolated from software because it's all part of the system that has to integrate together. There's certain functionality that's allocated to it, and in reality a lot of hardware is morphing more into software based on the extended use of firmware, and so all of these pieces need to tie together in terms of where are potential weaknesses, what could be the risks that we have to deal with, and how important are they to the operational mission. If something went wrong, is it critical, or can we work through it? I think those are all aspects that cybersecurity engineering needs to focus on.

What I want to share-- oh, did you want to say something else, Rita?

**Rita Creel:** Yeah. There was a second question from Brent about the-- when you have a hardware element that you have capacity requirements that are essential to doing any additional processing for security, the hardware teams have to understand that, have to understand that are there are impacts. So there's a lot of tradeoffs-- performance, capacity, security-- that all go into developing your final requirement set, and that's an important point.

**Carol Woody:** What I'd like to share with you now as one of these pieces is the importance of cybersecurity risk management, and we have found that that's a major area where tooling is limited. A lot of it is really more focused in terms of assembling and eliciting security requirements, defining what you need, with very little structure in terms of how do you think through the way in which problems could occur and the risks that you need to be concerned about.

As a result of that, we have developed a methodology that we will call Security Engineering Risk Analysis, or SERA is what we refer to it, and this is really a step-by-step approach to assembling a group of people with a range of knowledge about how the system should function and the potential concerns and really analyzing them early in the design so that you can begin to identify weaknesses, understand what potential risks can be there, and then figure out where there may be gaps in your requirements or gaps in your responses to security problems to address them early. So the hope with using this early in the lifecycle is that you build a better understanding of what you need.

So how does this work? Well, let me walk you through the steps. SERA is actually done in four steps, and the first one down here is looking at your technology environment, understanding, "How are we going to field this system?" What are the pieces, how they're going to work together, and what do we need in terms of how we expect this system to work.

Then we bring in the concern of a threat actor, and what could be here that could be exploited. This could be firmware; this could be ways the design is interfacing; this could be pieces-- what could break. This is the mindset of the cybersecurity engineer to look at the pieces and how they're put together. And if something breaks, what's going to happen? What do we care? Well, one of the key aspects we care about from security is the confidentiality, integrity, and availability of our data and the capabilities and the processing pieces.

So let's look at how these could be compromised if something would break that we would be concerned about, and also how would the attacker get to it. So that starts to look at the interfaces and the ways that we have potentially provided a conduit through our interfaces to allow some vulnerability that may be deep in our system to be accessible, and how would they exploit it. Once they do that, what would that mean in terms of our mission or operation? What results would that be? And then we can circle back and say, "How much do we care about this?" What can we do in our design, in our ways of structuring it, by changing the requirements, by adjusting how the pieces fit together, reducing the impact, potentially removing the adverse conditions and the mission impact, but ultimately improving how the system will operate through focusing on the mission we're trying to accomplish through the technology and where is it that we might need to be concerned about improving it. This gives us a way early on to look at requirements gaps and to look for the ways in which a system could break that we need to address.

Rita, I know you've been thinking about, "What do we do with this stuff?" Now we know how the system is going to break. Now what?

**Rita Creel:** So back to-- and I'll get to one of the questions with this slide-- back to the situation where we talked about the security requirements process needing to be more connected to the systems engineering and program requirements all together. So what this slide is showing is on the left we have our cybersecurity engineering, our risks, and we want them to be folded into the process of developing the program or the system's set of requirements, and that's going to involve tradeoffs with performance, with capabilities, with other types of quality requirements, and then with cost and schedule. So you have to come out with some way of indicating why these security requirements are important. We're going to get to that in measurement, but we have a question here from Leon: Do we not find that security requirements are relatively similar project to project? So why can't I just use the set of controls from one project to another? Carol, do you have thoughts?

**Carol Woody:** Controls are actually the response to a requirement. What you ultimately need to figure out though is, "Why do I need those controls? How can be system be attacked and what are the risks that I'm concerned about?" And yes, there are similarities in those that we're seeing. Actually, within the bounds of working with SERA, we're putting together some archetypes of attacks so that the engineers can then start to look at these and make sure that they're considering a range that we constantly see repeated when they're looking at their designs, in terms of how could it be attacked. Then ultimately when we determine what we want to do, that's when we select the controls. But it's equally as important to make sure that the controls are integrated in a way that they can't by bypassed, and it's not, in many cases, that the controls aren't there when we're engineering a system; too frequently there are ways that the attacker can bypass the controls so that they're not as effective as we need them, and that's a continuing problem we're seeing in the operational environment.

You mentioned measurement. Yes, measurement. We keep coming back to this. We don't have good measures but we do have ways of approaching measures.

**Rita Creel:** Yes, and one thing to do is to start with what's your assurance goal, and the broad one is kind of simplistic. It's just, "Well, the system is going to be resilient when under attack." I think that's the difference between the security analysis and many other analyses, is you have a concept of a smart adversary who's actually trying to break your system or steal data from it or take over or get it to do what you don't want it to do. So what you can do with that goal is decompose it into subgoals or assurance for various functions-- what are the critical functions for your system-- and then you can start to build measures from looking at those functions that need to be secure and resilient and what the different processes are that they implement, and then you can develop some measures from there. But again, you want to measure to answer questions, not

just to have a bunch of measures, and you want to be able to track your measures throughout the lifecycle to measure your performance against the requirements.

One of the questions that goes with this has to do with, I would call it, return on investment, and it's: Have you seen cybersecurity engineering translated into bottom-line contribution, instead of seeing it as an expense? And that's one area where you want to use measurement to assist with. You want to be able to say, "This capability, this security feature, is having-- it's reducing your risk in cost by this amount."

**Carol Woody:** There's also a question here from Valerie about: How is this different from software, safety, and vulnerability analysis? What we're finding is that there are a lot of similarities to safety considerations, but safety does not think in terms of an active, intelligent attacker. They think in terms of hazards of how the system might create itself problems, and what we are frequently seeing is that this active attacker can trigger unexpected safety considerations. So that becomes one of the aspects that needs to be looked at from cybersecurity engineering. Safety determines things that shouldn't be done and structures and environment that will operate safely, but it doesn't take into account the ways in which instability can be triggered.

Also in the area of vulnerability analysis, that's certainly an aspect of cybersecurity, but we find when organizations function and focus totally on that, they tend to run lots of tools, they generate tons of vulnerabilities, and then they have no mechanism by which to evaluate which ones are critical and how do we need to address them, except a generalized scoring system that doesn't relate it directly to the system they're actually building; whereas if they've done the right amount of risk analysis up front and really understand how their systems can break, how the pieces can fit together in terms of what needs to be an area of concern, then they can begin to sort through these vulnerabilities and better prioritize them for their own critical needs.

We've got one more area that I want to emphasize, and that is that this stuff has to be end-to-end in the lifecycle. Too frequently, like with that question of vulnerability analysis or safety analysis, we see requirements are really focused on good security requirements and good safety requirements, and then design really functions on (inaudible) vulnerabilities and nobody follows it through in testing, and nobody follows it through in implementation and deployment. So there really is not this focused, consistent view of responsibility for cybersecurity, so that ultimately in the operational environment you end up with what you need to function effectively with reduced risk. That's really what we're looking at. It's not that we've got specific new things that have never been done before that we're bringing forward. What we're basically saying is we need a new focus that really looks across all the pieces and ties them together, and then, with this measurement, allows management to then understand, "How well are we doing? How well are the pieces fitting together? Are we accomplishing what we need?"

I love this slide because it really focuses in on the issues of where the problems are showing up and where we're finding them, and you can see that we're finding a lot in design and development, but the bulk of this stuff that's showing up in early design and requirements is being missed. Our tools won't find them for us. This is where the engineering rigor needs to be focused on risk and cybersecurity, because we can't fix these at the tail end.

**Rita Creel:** Yes, and also, there's sort of a snowball effect. So one requirements defect or one missing security requirement can lead to several design defects or vulnerabilities, and then each of those design defects or vulnerabilities can lead to many more in code or in other aspects of your system.

**Carol Woody:** We have a question here from Drew that's an interesting one, and that is: Can you relate security measurements to a key system attribute such as resilience? Well, yes, we talked about relating it, but you can't focus just on resilience or just on reliability. Both of those are important, and even safety is important, but what you have to bring in is this risk perspective from an attacker's viewpoint to tie them together to ensure that you have sufficient resiliency, that you have sufficient reliability to make the pieces function.

One of the examples I've always seen is systems are designed with fantastic hardware reliability, but for backup, and then they run the same software on both the primary and the backup. So if they have a software glitch, guess what? Both the primary and the backup break. But too frequently systems engineers don't think of software as having aspects of risk. They think, "Okay, we've tested it. It runs. Nobody's changing the code, so we should be good to go." But in reality the way we interface pieces and put them together, the software is not necessarily executing exactly the same way every time, and that level of complexity and the combinations can cause instability frequently.

This is another way to think about how we need to look at the system and this gets to the aspect of where good cybersecurity engineering can be cost-effective, because if you really are spending the time and rigor up front, removing that 70 percent of defects I was showing you in the earlier slide, then it reduces what you have to deal with in the code so you don't have the snowball effect that Rita was talking about, and end up reducing what's actually implemented in operations; and we have data that shows that the early rigor and focus on quality engineering and cybersecurity supports this, but it does require the focus within the lifecycle to really have the right skills and resources to make that work.

Did we miss any questions? I want to make sure.

**Rita Creel:** Here's one, from Carlo: Are the risk analysis and management cycles responsive enough for dealing with the immediacy we often see in the operational field with regards to new threats and risks? Carol?

**Carol Woody:** I think they are, but it really requires somebody who's knowledgeable of those new aspects in the field. You can't just assume I studied operational security three years ago and I'm good. Well, the attacker's skills and capabilities are really mushrooming, and the way they are figuring out how to create havoc is increasing. So we have to be continually increasing our knowledge level. I'm constantly reading. I'm monitoring all kinds of activities just to keep my skills current, and I'm steeped in this. Your standard engineer who's focused on keeping the system up and running or figuring out how to make it work is probably not going to be the one you want to rely on to be really current on the most sophisticated issues that they may have to deal with. So this other set of eyes and this other set of knowledge needs to be there.

**Rita Creel:** And Carol, not only are the attackers getting more sophisticated, but we're constantly pushing out new code and new kinds of capabilities and new devices. So we're walking attack vectors with all our devices that the attackers are constantly--

**Carol Woody:** This year especially. Now that everybody's working from home, suddenly everybody's network has just suddenly expanded to include the world. Yes. Okay. Did we build our systems to do that? Probably not. So right now the operational environment is having to react to that, but as we build new ones, we have to recognize this is a reality of our world and factor that in. So that's, again, part of the learning process.

Some final thoughts. Start by building a cybersecurity strategy. This would describe the plan of how you want to go about considering it; the goal, like Rita talked about, in terms of if you're going to set up measures, what is it we're going to measure? Explicitly looking at the ways in which you're going to apply cybersecurity engineering. Who's going to handle this? How are they going to be responsible? How are they going to get access to all the information they're going to need across the full lifecycle? They can't just be isolated in a little corner; they have to be able to look at all the pieces and aspects. How are they going to consider requirements? What knowledge will they bring to the table? So all of these uses are valuable to start to assemble in a plan of, "What are we going to do?"

And then you work the plan, with the goal of measure, monitor, and improve as you move along, with the goal of reducing pieces, and we can tie it all together.

**Rita Creel:** One of the things that we've tried to emphasize is the need to focus early in the lifecycle. So on the left you see a mission thread or the business processes-- and again, as Carol mentioned, that is the initiating piece of our Security Engineering Risk Analysis approach, our SERA approach, where you're tracing through and trying to identify-- think like an attacker-- where are the valuable places for an attacker to come in, and you're doing that not just at the beginning; actually you're sort of carrying that mission thread through. I kind of like to see that blue shape on the left sort of continue through the process.

And another thing to remember is it's not just the system you're building, but the infrastructure that you're using to build your system-- so all of the software tools, the development environments, and if you've got hardware, the tools and environments that you're using to build your hardware. All of these are potential sources of risk that you need to be aware of and manage.

**Carol Woody:** We've got one question, which leads to our next slide, of: How can I get started? What can I do? From Patrick. I would recommend to you that-- we have books that you can read. We also have a certificate that you can take that focuses on five of the key areas that we've mentioned today, with requirements, risk, threat modeling, supply chain risk management-- all of these tying it together with the cybersecurity engineering perspective, and so that will give you pieces to leverage and it's building that mindset that we really recommend that you deal with.

We also have a website with a lot of other information that will be supportive to you, and I think we've hit most of the questions I can tell, and hopefully we've given you the steps that you'll need to get started, so I'll turn it back over to Shane to wrap us up.

**Shane McGraw:** I just wanted to say, Carol and Rita, thank you so much for sharing your expertise today and just a great job of sharing your years of work in this area and your expertise really showed today. We thank you for that.

We also want to thank our audience for attending. We had a worldwide audience. I saw Spain, Rwanda, Netherlands-- people staying after, obviously, their business hours to learn on this subject, so we just greatly appreciate all the great comments and questions in there and that's what makes these worthwhile doing. So we thank you for that.

We ask upon exiting, if you liked the content from today, please hit that Like button below your video window and share the archive with potential colleagues if you found value today. The URL for the archive will be the same as you're watching now. Also you can subscribe to our YouTube channel by clicking on the SEI seal in the lower right-hand corner of our video window.

Lastly, join us for our next webcast, which will be on December 8, and the topic will be Busting the Myths of the Programmer Productivity, with Bill Nichols, and we'll email everybody a registration link with that.

Any questions from today, please send to info@sei.cmu.edu. Thanks everyone. Have a great day.