Title: Identifying Software Security Requirements Early, Not After the Fact
Transcript

Part 1: Early Identification Reduces Total Cost

Julia Allen: Welcome to CERT's Podcast Series: Security for Business Leaders. The CERT Program is part of the Software Engineering Institute, a federally funded research and development center at Carnegie Mellon University in Pittsburgh, Pennsylvania. You can find out more about us at cert.org.

Show notes for today's conversation are available at the podcast website.

My name is Julia Allen. I'm a senior researcher at CERT, working on security governance and software assurance. Today I'm pleased to introduce Nancy Mead who leads CERT's research efforts in software security requirements engineering. We'll be discussing why it's important to identify security requirements early in the software development lifecycle and how to get started. So welcome Nancy, glad to have you with us today.

Nancy Mead: Oh thank you. I'm glad to be here.

Julia Allen: So some would say that we're already spending a fortune on software development. So why should we spend time and money on developing security requirements before we have a system in place, and can we really justify the additional expense?

Nancy Mead: It's true that we are spending a lot of money on software development, and I could understand why people would have this kind of question. I think that a lot of them don't realize just how expensive it is to fix requirements problems once a system is in production. Some studies show that it costs 10 to 200 times more to fix a requirements problem after the fact. And in looking at overall software project effort, re-working requirements defects can account for 40 to 50% of the total effort. It's really a large percentage. So I think it's really important to pay attention to requirements early on.

When we think about security requirements, I don't think we really have much of a choice. It's pretty clear that what we're doing to improve security right now is not adequate. Many of our systems, including our critical systems, are pretty vulnerable. There's so much software present in all of the systems that we use that we have to start to address security in a systematic way.

Organizations that want to maintain their reputation, satisfy the concern of clients, and reduce the amount of effort spent issuing patches or combating break-ins will have to address security early in the lifecycle.

For example, over the last few years we've seen a major effort by Microsoft to address security in some of their Windows products. When a large organization like Microsoft thinks that security is important enough to start addressing it early in the lifecycle, I think that people ought to pay some attention.

If you don't address security requirements early on, there's a good chance that you'll make architectural decisions that will make it very difficult to add security later.

Julia Allen: So Nancy you said that you've got these effects or impacts — if you address a software requirements defect later in the lifecycle, it's much more costly. But isn't it the case that someone who's managing a software development effort may not have responsibility for the whole total cost of ownership? In other words, they're responsible for getting the product built but they may not be responsible for fixing the defect once the system is in the field. So is there some way to get the argument that you made about full lifecycle cost and wanting to address the defect early in the lifecycle, is there a way to kind of make that argument for someone who doesn't have the whole lifecycle responsibility?

Nancy Mead: If you don't have the whole lifecycle responsibility, then I think you need to look at it a little bit differently. Perhaps you need to think about your organization's reputation. If you put out a product that has a lot of problems then that's going to hurt your reputation in the field.

And I think the other thing that is perhaps needed is really good management support. What we find is that software development folks can't really do these things in isolation. They have to have, at some level, management support in order to make additional investments. Because it's true that the investment that you're making in security early on won't necessarily result in benefits while the software is in its development part of the lifecycle. The benefits will come later when the software is in operations.

Julia Allen: Okay, well that makes sense. But if I'm scanning my code for vulnerabilities and if I'm doing regular penetration testing, and also if I'm using things like password protection and firewalls, aren't these practices sufficient to protect my software from attack?

Nancy Mead: One of the things that I've observed is that people tend to use a standard list of security mechanisms, such as these.

For example, we had one client that we worked with that had a very nice functional requirements document, and then at the end they tacked on an appendix that was just a list of standard security mechanisms, such as the ones that you mentioned. And the problem with that is that it's not really tailored to the system at hand, and in fact these mechanisms aren't adequate to protect. If they were, we wouldn't have some of the security break-ins that we do have.

So I think that organizations tend to change the way that they're thinking about these things so that they do actually consider the security requirements for the system at hand, and not just try to add on some mechanisms, such as encryption or password protection or firewalls. Those tend to be thought about very late in the process, and it's really sometimes too late to get the kind of protection that is really needed for these systems.

I'm very concerned when I see an organization that only has a list of these mechanisms because it shows that they haven't really thought about security. They know they ought to be concerned about it and they're just throwing this stuff on as an attempt at a response. But by and large it doesn't really work.

What I compare it to is one organization that when we asked them why they were using a particular email application, they said that "Well all of their office products were from one vendor so they decided this particular product may as well be from the same vendor." And that was the level of tradeoff analysis that they did.

It's the same thing with requirements. If you don't think about them in a deep way, you're not really going to be able to do an adequate job. So from my viewpoint, the mechanisms that you discussed are necessary but not sufficient to address the problem of security.

Julia Allen: Right. It also seems to me, based on our work, that you can put some of these mechanisms in place late in the lifecycle or after the fact as kind of a reactive response. Either something's happened and you put these in place, you put new firewall filters in place, or you notice that that's what the marketplace is suggesting. But again you're in this reactive, kind of catch-up, kind of try-and-stem-the-tide mode. You're really not getting in front of the situation at all.

Nancy Mead: Exactly. It's analogous to what we've seen in the safety area. If you look at something like the space shuttle, for example, the safety mechanisms that they have built-in occur all along the line, not just at the end, and also not just in the beginning. So I think we need to take that kind of holistic approach with security as well.

Part 2: How Are Security Requirements Different?

Julia Allen: Well that's a nice lead in to my next question which is how are security requirements different from other types or kinds of requirements?

Nancy Mead: I tend to look at it in terms of how requirements are identified and developed. Many requirements engineering techniques that are in use today, when people do requirements engineering in a systematic way, are aimed at functional requirements, what I would call end-user requirements. So they tend to focus on user features and things that the user can do with the system, but they don't do a very good job of highlighting or capturing other kinds of requirements – for example, quality requirements in general, such as maintainability, availability and so on, and security requirements in particular.

What we've found when we go into organizations is that they need to use a different kind of approach to identify security requirements at all. The security requirements that they do identify, when they finally get around to it, tend to be a combination of both functional and non-functional requirements.

For example, if you're implementing an access control mechanism in the code, that in fact does have some functionality and you will need functional requirements that correspond to it, but they may not be what I'll call end-user requirements. It may not be something that is really visible to the user. In addition, you have non-functional requirements in the security area and those need to be handled as well.

So my finding is that if you just go along and develop requirements, business as usual, you'll probably overlook quite a few of the security requirements that are going to be needed because it's not what comes to mind. Very few users are going to put security requirements at the top of their list of needs because it's just not a user-oriented feature.

Julia Allen: Right. In fact I've found that one of the key aspects for addressing security requirements is thinking like an attacker, and most developers and software engineers don't think like an attacker – right? – in terms of what the software should or should not do under attack?

Nancy Mead: Definitely. And certainly the users don't think about that either. That's far from their mind. They're worried about the features in the system that's being delivered, not in whether or not somebody might be in a position to attack it. And definitely, for the most part, developers don't usually think about attack, because usually, as you know, when you're developing software you have enough work on your hands without worrying about what might be viewed as something extra.

Julia Allen: So in addition to all the things that we expect our software developers to know about, what knowledge and skills do they need to add to their bag of tricks, if you will, to be effective in developing security requirements?

Nancy Mead: At first you might think that you need a whole lot of extra skills, but I think you'll find that a lot of the skills are already present on the team. But they may not need to be present in each one of the members. Not every software developer needs to worry about all of the security aspects. You do need a team member that has security expertise and if that person with the security expertise also has good knowledge of software engineering, that's ideal. But it's possible that you can take a very knowledgeable software engineer and somebody who has a good background in security, and between them they can do a lot to identify the needed security requirements.

Julia Allen: Are there specific points of view or perspectives? Or kind of when you take a knowledgeable software engineer and a knowledgeable security engineer, what are some of the insights perhaps that they might gain from one another to create useful security requirements?

Nancy Mead: Well, for example, the software engineer will tend to have more of an understanding of how the system is going to be used in the end and they'll tend to think of it more from a user's perspective. Software engineers, for example, can be very good at developing use cases, but they tend not to think so much about the security aspect; whereas the security specialist may think about security to the exclusion of how easy the system is to use. And so both of those skills are needed in order to both identify a set of security requirements and to understand what is reasonable. It's possible to specify, design, and develop systems that are completely secure, but if they're so secure that they're extremely burdensome to use, then there's no point.

Julia Allen: Right, it's like they say when you're talking about attaching your systems to the Internet, if you wanted to be 100% secure you wouldn't attach in the first place, right?

Nancy Mead: Exactly, exactly, that's an excellent point.

Part 3: Getting Started

Julia Allen: So if I wanted to get started – if I was a software project manager or if I was responsible for a software development effort, and I wanted to get started in defining and laying out a process for software security requirements, what are some of the first steps I might take?

Nancy Mead: What we've found is that one of the best first steps is to try to get a good understanding of what it is you're doing now, in software engineering in general. What are your current software engineering practices? If you have a good understanding of that, then you can start to look at how to add a good security requirements engineering process to that overall mix.

One caveat is if you don't have a good software engineering process at all, it's going to be very hard to excel in security requirements engineering because then you're just applying a point solution. We found that if you try to improve in one isolated area, there's a good chance that it won't carry over to the rest of the software development process, and after you've done it once or twice, probably people won't be able to maintain that level of involvement because if the rest of the process is chaotic, improvement in one area isn't likely to survive.

So I guess the first requirement is that you have a good understanding of your current practices. Presumably you actually have current standard practices and you add security requirements engineering into appropriate places in that process.

One of the processes that we've developed to address security requirements is called SQUARE, and it stands for Security Quality Requirements Engineering. It's a 9-step process. It allows a team of requirements engineers, developers and stakeholders to systematically go through this process and actually define security requirements. We're finding some pretty good success with this in the field.

Although one of the things that people want to know is "Well how can I integrate this into my current set of processes? I don't want to start all over again with something else." And so we're starting to address that as well.

But we do recommend that whatever process people use, that they approach security requirements in a systematic way - that they go out and benchmark other organizations that are doing a good job in this area; that they look at the literature and start to systematically incorporate an approach in-house that can be helpful to them.

There are several areas that are really big in terms of getting started. And one is really simple but you'd be surprised how often people don't do it, and that is to have a common understanding of what you mean by some of these terms that we throw around.

I worked with one group where there were nine stakeholder representatives, and each one of them had their own idea of what was meant by availability. Well if you start out with all of those different viewpoints, it's hard to imagine that you're going to arrive at a common set of requirements because you don't have a fundamental common understanding of the basics.

Another area where organizations could focus on, very early on, is to decide upon and document their security goals. If you don't have good goals, it's hard to understand how you can develop good requirements. And again, we've gone into organizations where we asked to see their security goals and they don't really have any. They might have a business goal for the system that they're developing but they haven't really given any thought to their security goals.

A third area that is key is doing a good risk analysis, including a security risk analysis. Many organizations will do a risk analysis but they tend to focus on things like cost and schedule risk and security really isn't part of it. So I think that doing that sort of risk analysis, which is something that many organizations already do, and just adding security aspects is going to give them a big leg up in terms of identifying security requirements.

So none of these things are really hard. It's just something that takes a little bit of extra attention.

Julia Allen: Well it sounds like with some of the recommendations you've made, these are just good tried and true things to do, regardless of the system or the software that you're trying to develop, and it sounds like you're trying to bring security to the mix as part of some fairly standard conversations like goals and risk analysis. Is that a fair statement?

Nancy Mead: Yes definitely. We find that when you get into it a little further there are things that need to be done a little differently for security. For example, when you elicit requirements, if you use an elicitation technique, very often they're oriented toward functional requirements, and we have found that some elicitation techniques work better than others if you're trying to focus on security.

Similarly we've found that people generally don't have enough funding to implement all of the security requirements that they could implement, and so we've found some good approaches for prioritizing security requirements as well, looking at it from basically from a cost-benefit kind of a viewpoint.

And so there are some specific techniques that are proving to be very useful in the security area, but the steps that you go through are not that different from what you would do from your general functional requirements.

Julia Allen: Well Nancy, you've laid out an excellent, I think, introduction and beginning set of thoughts and suggestions for our listeners. Do you have some sources where you can point our listeners for more information?

Nancy Mead: Fortunately there's quite a bit out there. The first source that comes to mind, of course, is the book that we've just written, *Software Security Engineering: A Guide for Project Managers*. It has an overview of security requirements engineering, in addition to a number of other topics.

A second source, which provides more detailed information, is the Build Security in website that's sponsored by Department of Homeland Security. And so that's an additional source that provides information on security requirements engineering, including some of the details of the SQUARE method and the associated case studies.

Finally, the CERT website and the SEI website both have information on requirements engineering. The SEI website has all of the reports on the SQUARE case studies, the SQUARE method itself, a method for comparing various security requirements engineering approaches; and for SQUARE in particular, we have a report that describes how to integrate it into existing lifecycle processes such as the Rational Unified Process, the Spiral Model and others.

Julia Allen: Well Nancy I'm so appreciative of your time and expertise today, and thank you for speaking with our listeners about software security requirements.

Nancy Mead: Thank you Julia, it's been a pleasure.