Building More Secure Software
Transcript

Part 1: Software Security Is Just Good Business

Bill Pollak: 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 more about us at cert.org.

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

My name is Bill Pollak and I'm the Manager of Communications for the SEI. Today I'm pleased to welcome back Julia Allen, a senior researcher with CERT and a recognized leader in security governance and software assurance. We'll be discussing why security is a software issue and how to address it. Welcome, Julia.

Julia Allen: Well, thanks Bill. I'm really looking forward to exploring this topic with you today.

Bill Pollak: Well, so am I. Generally, we think of security as an operational IT issue focused on defending our computers and networks from attackers and from security breaches. Or we think of security as information security, concerned with protecting information in digital form. But today we're discussing what you call software security. So what is software security and how is it different from IT or information security?

Julia Allen: Well, Bill, obviously in much of our work at CERT and in the software and IT community at large, we're all very familiar with IT security, defending networks, defending systems. With all of the compliance issues that are emerging today and protecting personally identifiable information, we're all becoming much more aware of information security. But software security is kind of just starting to make it onto the radar screen, and what software security, in a nutshell is, is building better, defect-free software, because typically software has many defects, many vulnerabilities and quite a few of these tend to be the source of security vulnerabilities that show up in our operational networks. So another way to think about software security is that it's software that is more able to resist attack. But in the face of an attack – a successful attack – it's better able to tolerate the attack and recover from the attack as quickly as possible.

So to net it out, it's less vulnerable to attack, a little bit more bulletproof and some of the ways that thinking about software security is different than IT or information security is, when you think about IT security it tends to be more detect and respond, finding and patching vulnerabilities; it tends to be more remedial or reactive. Whereas software security, the idea is to get in front of the issue by building more robust software. You address the issues much earlier in the software development lifecycle, it's more preventive or more proactive, if you will, and attempting to actually put better products in the field so we hopefully can eliminate some of the issues we're dealing with today.

Bill Pollak: Okay. So why would you say that business leaders need to pay more attention to software security?

Julia Allen: Well, there're probably two major, I would say, trends or topics that are worthy of business leader consideration. The first one is that attackers are getting much smarter and the second one is it's just good business. And I'll say a little bit more about both of those.

So what we're seeing is that the attacker community is becoming much more sophisticated in their approaches. In fact, it's becoming actually a well-funded profession with a very robust underground economy where sensitive information, identities, credit card information is being bought and sold quite freely. And what's happened is, as a community we've gotten better at protecting our networks and our infrastructures and our information. Attackers are [now] focusing on the more vulnerable parts of our systems, which are the applications, particularly web-facing applications. And they're the primary gateway to sensitive data, and so that's where software security can really provide an advantage. So that's the part about the attackers becoming a bit more sophisticated and kind of upping the ante for all of our attention. Oh, another thing to think about there is an attacker really only requires one entry point into a system where as software developers and IT staff need to anticipate and defend all — which is a near impossible requirement. So that's one aspect of why business leaders need to pay attention.

In terms of it being just good business, we're seeing more and more indicators that the market place is starting to demand it, particularly from organizations that develop software. They're saying, "We want secure products and we're not going to put your product into our infrastructure unless it's secure." The total cost of ownership for software, 50 to 80 percent depending on whose numbers you pay attention to, are in the maintenance and operations of software. So that's where a huge amount of the cost is going because of the quality of the software that's going into production. What we've found is anywhere upwards of 50 percent of operational software vulnerabilities are actually design flaws that could have been discovered much earlier in the lifecycle. And in fact, you can save anywhere from 100 to 1,000 times cost and schedule, in some instances, where it's more cost effective to identify and correct the defect early in the lifecycle versus later.

So I think, to net it out, we really can't keep up or get ahead of the curve by dealing with security as strictly an operational issue. So we just need to find a way to get better products in the field.

Bill Pollak: Great. How did software get to be so insecure? Why is software so insecure?

Julia Allen: Well, certainly being at the Software Engineering Institute, you know this and we've seen many, many instances where software is just growing in complexity, we keep adding to it, we keep changing it. Our market places and our various customers want new features, new functions, new services, faster. And more software means more vulnerabilities.

When you compound that by the fact of how connected we are today, in terms of our Internet connectivity, the fact that development is happening globally around the world in all kinds of places and trying to be assembled and put together. The fact that we're always adding new components, new pieces, we're interfacing a very complex system with other complex systems and so we have what we call systems of systems issues.

And then that's further exacerbated by more and more third party software where we're, perhaps, maybe we as an organization have responsibility for putting the whole system together but we're not developing most of it ourselves. And so we have to apply the same rigor and requirements and process and service level agreements on our suppliers as we do on ourselves.

So, you think about all these different factors, it's really hard, even for a very talented development team to get their heads around these issues and try and figure out how to tackle all that complexity. So you throw security into the mix and it's a pretty daunting undertaking.

Part 2: Develop Software with a Security Mind-set

Bill Pollak: Well, how is developing software with security in mind different from normal software development?

Julia Allen: Well, this is probably kind of the crux of the key message I want to get across to our listeners today which is how to think with a security mind-set. So I would say there are just a couple of factors to point out here and that is to think about security from the beginning of the lifecycle. Security, at least in its history, tends to be an afterthought, if it's thought about at all. And what we're finding in trying to tackle this as a software security engineering issue is you really need to think about it from the very beginning, when you're going through acquisition, when you're going through requirement specification, all the way through the lifecycle.

Typically how we've thought about other, what we call, non-functional characteristics or attributes like performance or reliability. Think of it during requirements, during design, during architecture and all the way through implementation, test and deployment. So that's one facet that's a little bit different.

Probably the toughest one for software project managers, architects, designers and engineers is they need to learn to think like an attacker. We often think about what the software should do, what the functions, features, and capabilities are, but we rarely think about what the software should not do. And I would hazard a guess that we really don't think about it much, what it should do and not do, when it's under attack. So being able to fold that kind of thinking in throughout the lifecycle.

And certainly last, but not least, security as a risk management issue and that extends all the way through when you think about security during the development lifecycle. So constantly assessing, at each lifecycle phase, what the highest areas of vulnerability and risk are, addressing the highest first, and then fully recognizing that as you go through the lifecycle, it's probably going to change quite a bit, depending on what kind of applications you're building, what kind of services that you're providing. And so the risks and priorities for tackling them need to be assessed and they will change over time as you go through your development.

Bill Pollak: Right. So maybe you could give us an example for some of the known good practices for developing more secure software. I know you can't cover all of them but maybe a few.

Julia Allen: Well, that's the good news. The good news is that there is an emerging body of knowledge and experience and practice that organizations who build and buy software are starting to identify. So, as I mentioned earlier, probably the first key one is to take any of these practice ideas and integrate them with your existing development lifecycle. So we're not talking about doing something – adding a whole new process or a whole new development strategy or method or lifecycle definition. But take your existing lifecycle and take these practices and add them to it.

And so some of the key practices that we've found particularly useful, during requirements engineering and some of the architecture design work, a practice called misuse and abuse cases. And those are helpful when you're adding a new requirement or a feature into your requirements, give some serious thought to how the feature could be unintentionally misused or intentionally abused by an attacker. For example, assuming if you have a requirement to interface a web server and a database server for some type of user-facing application, often a developer will assume that the connection between the web server and the database server can always be trusted. And so, when you have a misuse or abuse case, you can actually challenge this and come up with scenarios and various ways to exercise, assuming then that the interface is untrusted.

Another example of some good practices would be the development of what we call attack patterns, where you actually capture a class of vulnerability, how that vulnerability can be exploited, what kind of attacker skill is required. And you can use attack patterns throughout the lifecycle to say, "Okay, if I were an attacker, how would I break this design or how would I infiltrate this particular piece of code?" Probably some of the more mature practices are in the secure coding area where you could – there's lots and lots of great guidance on secure coding practices for different languages, using code analysis and other scanning techniques.

And certainly last, but not least, all kinds of great approaches to security testing, white box and black box Testing, and using something called threat modeling where you can take this attack pattern idea and actually fold it into your testing suite and kind of the tried and true penetration testing can also be very helpful.

Bill Pollak: Very good, thanks. So what are some of the effective ways that folks can use to get started?

Julia Allen: Well, obviously we can't boil the ocean on the first day. And so we have to think about taking small steps from the beginning. Probably one of the best ways to get started is to take a look at the competencies and skills of your current development team and enhance them with some security experts. The organizations that have had great success tackling secure software development have actually put security experts living on, co-resident with the teams, helping guide them and educate them and inform them as they go through their development.

Clearly, the highest return on investment is to try and tackle or add security practices as early in the lifecycle as possible because you get the greatest benefit. But adding practices early in the lifecycle probably isn't as mature of a practice, like during requirements and elicitation, engineering and maybe during the early stages of architecture. So most organizations today are actually starting out with secure coding, looking at secure coding practices, code analysis, lots of peer review, doing various types of lower-level testing to tease out some of the defects that might have gotten through the code. And this is the most mature set of practices that are in use today and so a lot of organizations will start there.

Independent of any particular practice, in all of our improvement initiative work at the Software Engineering Institute, there's some kind of tried and true ways to get started. First you have to understand what you're trying to do. Why do we care about secure software engineering? Why are we doing this? What are we hoping to accomplish? And so make sure the incentives are clear. Get good buy in from your sponsors so you can have a sustainable improvement initiative. Find some early pilots, just a small number, and make sure to demonstrate some early results. Make sure people understand this is going to take time. Like a lot of things in software, it's a never-ending journey, but because of the threat and requirements and risk landscape is always changing. We always say "communicate, communicate, communicate," right? That's you're line isn't it? Lots of awareness training and ongoing education.

And although putting secure software engineering practices in place is always much easier on a new project where you have kind of a clean slate to start with, we're getting some good case studies from the field that actually rolling up your sleeves and tackling the harder issue of legacy systems – systems that are already deployed, maybe where you have a lot of third-party software and actually subjecting that software to some fairly rigorous scrutiny may be more challenging but produce the highest payoff. So those are just some ideas for getting started.

Bill Pollak: Thanks, Julia, this has been enlightening. Where can our listeners learn more about software security and software assurance?

Julia Allen: Well, there's some excellent work that we've been participating in with the Department of Homeland Security Software Assurance Program. Over the last couple of years, we've helped them stand up, with many other contributors, the Build Security In web site.

We've had the good fortune to take a lot of that content and actually put it into book form. So we have a new publication, a new book called Software Security Engineering: A Guide for Project Managers, which is now a book in the Addison Wesley SEI Series as well as the Software Security Series.

And both the BSI web site and the book have a number of excellent references, both for project managers and software developers, so I would point our listeners there.

Bill Pollak: Very good. Well, thanks for being with us today Julia.

Julia Allen: Well, Bill, this has been great and I look forward to doing it again.

Bill Pollak: Thank you.