Mainstreaming Secure Coding Practices
Transcript

Part 1: Making the Business Case for Secure Code

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 welcome Robert Seacord, the leader of CERT's Secure Coding Initiative. Today Robert and I will be discussing why secure coding practices should be on a leader's radar screen. And more importantly, how to get started including taking advantage of some great, emerging, new standards. So welcome, Robert, really glad to have you with us today.

Robert Seacord: Thanks. It's good to be here.

Julia Allen: Okay, so let's start with the hard question first. How do you make the business case for secure coding practices, particularly given that there are often much more pressing business demands? And certainly today's economic climate, how do you get that point across?

Robert Seacord: Well, I don't really think that's that hard of a question. It's sort of similar to what we're typically asked in the software engineering field. Because typically, making code more secure is roughly equivalent to making it higher quality. And generally speaking, building software better tends to reduce the overall development cost.

So the problem with insecure code is it does have a cost to both the software development organization and to the customer. So to the software development organization, any kind of defects, particularly post-deployment defects, are expensive to repair, and especially things that are resulting in publicized or vulnerabilities that are actively being exploited. Because now you need to respond in sort of an emergency manner which drives up the costs of providing these kind of emergency repairs.

And of course, there's always a reputation cost. There was a study at CMU a few years ago that showed a drop in stock prices related to publically disclosed vulnerabilities. And frequently there's even a higher cost to customers who, well could have losses due to attacks resulting from these vulnerabilities. And also costs related to just managing and installing the patches, even if they don't come under attack by some kind of hacker. In a way, they're under attack by their supplier.

So again, I think that developing code well, developing securely, actually can reduce development cost. One of the things we're doing is working with the SEI's Team Software Process. And they have data showing 78 percent improvement in productivity to go along with the quality gains.

Julia Allen: It clearly makes sense if an organization is developing software products to make their products more secure. Because, as they have to support them in the field, if they haven't done their due diligence and put the right practices in place, they have a long maintenance tail. But what I also hear you saying is as a using organization, if you're using someone else's software, clearly there's a cost there too, right? So both on the developing and the using side.

Robert Seacord: That's correct.

Julia Allen: Okay, so if we accept the premise that you've just laid out — that developing more secure software is important as a part of just doing our job building quality software — how do we decide what part of the life cycle, the software development life cycle, to tackle first? I mean, clearly you can start at the very beginning, requirements engineering, or you could start with scanning for vulnerabilities. How do know where to focus your attention?

Robert Seacord: Well, it is important to understand the requirements of your system and what sort of information you want your various users to access. So if you don't have that understanding, of course, it's impossible to develop a secure system. And you need to implement secure designs. Without secure designs, again, secure coding practices can't help you.

The back-end stuff, the static analysis tools and other tools for assurance and testing, often become a focus but really, in my opinion, tend to be overly relied upon. And the reason is that you want your developers to understand what the issues are and to develop code securely the first time. And that's because when they're initially writing it, they understand the context in which the code's being written, they understand the complexities of the software. And that's the time to get it right.

So if you wait until later in the development cycle, where you've got a static analysis tool that's finding some, reporting some defects, and those are getting logged into some sort of defect tracking system, and being assigned back to the developers, now they've got to re-familiarize themselves with the code and make some repairs. And at this point, the cost has gone up and the quality potentially is going down.

So static analysis tools I think are important as a quality assurance step but I think it's more important to code correctly the first time. And that's really one of the tenets of the Personal Software Process (PSP) that Watts Humphrey has developed.

Julia Allen: Well I've been thinking about this as part of our software assurance work. And we see a lot of energy in this, as you said, the latter part of the life cycle, the testing and the diagnostic piece. Because there are tools, it's more prescriptive, it's easier to get your hand around. But I think you very quickly get buried in minutia, right? You've got these long lists of vulnerabilities and really no way to address them. And as you said, the developers then have to go back and refresh on how to tackle them, right?

Robert Seacord: And it's actually worse than that now that I just reflect on it a little bit. Because very few of these tools really perform verification. So they're able to identify some violations of secure coding rules, meaning that just because you've got a clean health bill from one of these tools doesn't mean that your code is free of security defects that could lead to vulnerabilities.

Part 2: Working with Software Suppliers; Selecting the Right Coding Practices

Julia Allen: Well given that so many business leaders find themselves in the position of buying versus building software, what are some of the important considerations when they are working with a supplier to develop or integrate software for them?

Robert Seacord: Well, the most important thing is understanding your requirements and what it is that you actually need, and once you understand what you need to remain engaged during the development process. So not to just throw the requirements over the wall and sort of disengage.

So one possibility is just to actually prototype the system to figure out what your requirements are. It's more of a collaboration than it is just strictly you purchasing something. One of the things we've also done here at CERT is we've developed or are developing secure coding standards for commonly used programming languages, such as C, C++ and Java. And these provide a baseline of correct coding practices that you would want to see enforced in the software that you're purchasing. So one option is to require conformance with a secure coding standard, either something that's developed here at CERT or developed elsewhere.

Julia Allen: Are you seeing any uptake on that idea yet — where perhaps either a standard or a subset of the standard is being required as part of a contract or a service-level agreement?

Robert Seacord: There was actually just an article about New York State requiring their software developers to conform or address the programming errors in the SANS/CWE Top 25 top programming errors list. So that sort of thing is starting to occur.

And I also just had a lengthy conversation with some people involved in international standards development. And actually, one of the concerns that that audience has is that these kind of coding standards might be in the forefront to creating regulation.

Julia Allen: Well, that's a real nice segue to something I definitely wanted to ask you about because it seems to be a hot topic right now. There are a number of efforts. You mentioned SANS, and then there's the OWASP, the Open Web Application Security Project, that are generating these top 10, top 25 lists of good software security practices specifically focused on reducing code vulnerabilities. And I know you've been involved in some of these. What do you find are some of the pros and cons of using these types of lists to drive your efforts?

Robert Seacord: Well, I think they're beneficial in that they can raise awareness of critical issues. But typically, just taking a top 10 or top 25 list is insufficient. So the CERT C Secure Coding Standard, for example: We have about 200 guidelines, which are all prioritized according to what the consequence of that rule being violated, how likely it is that a violation of that rule can lead to a vulnerability, and remediation cost for people who are addressing legacy code issues.

So it probably makes more sense for an organization to look at a complete list of issues that they need to address and then try to cut that down according to their particular application and their requirements.

Julia Allen: So how do you know when you're tackling these lists how much is enough and when to declare success?

Robert Seacord: Well, it's pretty tough. You know, the issue is always that all you need is one vulnerability to have a fairly significant problem. So I think it's just a question of thinking about how your application is being deployed, what the risks are, and trying to initially focus on those sort of coding issues that can lead to high, critical exploits.

Julia Allen: I appreciate you making the link to risk. Because I think that given that we know we can't do everything, that at least gives us one approach or method for helping us rank, stack, and prioritize where we need to pay attention.

So you had mentioned, Robert, that these emerging standards are starting to look very promising. And of course, you've been leading CERT's efforts in that regard. So could you say a little bit about the efforts that you're involved in and which ones you think are the most promising?

Robert Seacord: We're trying to effect changes in language standards to make these languages more secure by design. So in particular, we're involved in the INCITS, the PL22, which is the committee that has oversight for all the programming languages. And PL22.11, which is C programming, and .16, which is C++ programming language.

So INCITS is the International Committee for Information Technology Standards. And it's the primary U.S. focus of standardization. All those committees have corresponding committees in ISO/IEC. And so INCITS committee basically establishes the U.S. position for the ISO/IEC committees.

We're also starting work on a language annex for the C1X, which is the next major revision of the C standard, where we're trying to create an informative annex which would allow compiler vendors to provide a version, a flavor, of their compilers which can be used to code more securely. And if that takes place, I think that's got a huge impact on the industry because basically we can make C and C++ programming as secure as Java programming. And so the only reason we haven't so far is just sort of a lack of will.

And if these changes are made through the standards process and in specific implementations such as GCC and Visual Studio and so forth, all that's required then is that source code is recompiled. And development organizations will automatically pick up a lot of protections that are currently lacking in existing applications.

Part 3: Starting Up a Secure Coding Project; Barriers and First Steps

Julia Allen: What do you find to be the biggest sticking point? You said, "We know what we need to do. What we're lacking is the collective will." What do you find to be the biggest hurdle or barrier to helping folks get started?

Robert Seacord: Well, the biggest barrier to security is not getting them started, it's performance. So I'll give you an example, a conversation I just had with Microsoft where they're planning on removing some options, some default options, from their compiler which would make the resulting code more secure. And the reason they're removing these options is because they've had push back from developers who are unhappy with the resulting performance. And generally we're talking about a two or three time slow down in performance.

So part of this is coming up with solutions that have adequate performance and not trying to force overbearing solutions on developers. But part of this has to be developers being willing to accept some performance hit in order to drive down the number of vulnerabilities that they're deploying in their software.

Julia Allen: So pragmatically, if a business leader wants to get started, if they would like to start a secure coding initiative, who do they put in charge and what are some first steps that you recommend?

Robert Seacord: Well, I think whoever really has responsibility for software development in that organization should be the person put in charge. I don't really believe in forcing solutions on people. I think people should see the value of the solution and want to adopt it.

I think the first step is training and trying to produce awareness within the development group. Because generally speaking, I don't think engineers set out to develop insecure code. I think they're just doing it because they don't understand some of the risks and some of the vulnerabilities and some of the potential errors they can make that aren't being diagnosed by their compilers and by the tools they're using.

Another good step, again, is to adopt a secure coding standard. And that just provides an organizational standard. So without adopting any type of coding standard, basically you're leaving it up to the individual developers to determine what are the set of standards to which their module is going to be developed. And that results in very sort of inconsistent code quality across the system.

Julia Allen: And what do you find on the training front? How long does it typically take, just ballpark, to get a fairly competent software development team properly trained in secure coding?

Robert Seacord: I think it's important to at least establish an awareness of what at least some common errors are and how those things can be exploited. And that just will get your developers to start thinking about what sort of code might put them at risk.

At the SEI, we have a four-day course in secure coding in C and C++. And that pretty adequately covers the basics. But you have to use that as a starting point and then make sure your organization has the means to keep up and track the evolution in software security and secure coding practices.

Julia Allen: Do you have some sources, including your own work, where our listeners can learn more on the subject?

Robert Seacord: Sure. We have a couple of secure coding websites. One is a wiki where we developed our secure coding standards and those are actually developed as a community process. So different organizations can get involved and contribute to that effort. And I've written a couple books on this topic.

Julia Allen: Well Robert, I so appreciate your time and your expertise today. The work that you and your team have been doing to push forth and stay the course on the standard I think really serves our community well. So thanks very much for your time.

Robert Seacord: Okay, thank you.