The Future of Cyber: Secure Coding

Featuring Steve Lipner as Interviewed by Bobbie Stempfley

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Bobbie Stempfley: Good afternoon. My name is Bobbie Stempfley, and I am the CERT director at Carnegie Mellon University Software Engineering Institute. It is my pleasure to have you here today for a podcast with one of the pioneers in cybersecurity, Steve Lipner. He is currently the executive director of the Software Assurance Forum for Excellence in Code, known as SAFECode, a not-for-profit organization dedicated to increasing trust in information and communications technologies, products, and services through the advancement of effective software-assurance methods.

Before he retired a few years ago, he was a partner director of software security at Microsoft and the creator and longtime leader of Microsoft Security Development Lifecycle process. He is also the chair of the U.S. government's Information Security and Privacy Advisory Board.

You are a very busy man. Thank you for joining us today, Mr. Lipner.

Steve Lipner: Steve is fine, Bobbie, and thank you for having me. It is a pleasure to be here.

Bobbie: It is really great to be reconnected.

Steve: Yes, indeed.

Bobbie: I find the culture of computers and information technology changing so rapidly today that sometimes we find it difficult to get our grounding, and you have always had your grounding. This podcast is really about exploring the lessons of the past and how we can apply them to the future, so that we are not recreating problems. I think you have got such an important story in this space. You were hyper-focused on software before many others did. So I really look forward to hearing from you today about the kinds of issues that you have been working.

Steve: Thanks very much, Bobbie. It is a pleasure to be here. As you say, or as you suggest, I have spent a long time worrying about different instantiations and manifestations of *How do you build secure software?* And we finally got into a position and an approach that we think actually made a difference, and I am really happy to have had that happen in time for me to see it come to the fruition it has.

Bobbie: I want to spend a little time exploring that. But first, can you tell me about SAFECode? What are the problems the organization is working to solve?

Steve: SAFECode has been around since about 2005 or 2006, and it is a nonprofit, it is an industry nonprofit. The members are companies that have to deliver secure software: big ones, small ones, online-service companies, packaged-software companies, and some that are manufacturers that include software in the devices or products that they build. So quite a wide range, but everybody bound together by this challenge: *How do we deliver secure software? How do we deliver secure software better? How do we get the software we depend on to be secure, so that we are not challenged by supplier problems?* And *How do we enable the rest of the ecosystem in the industry to understand what effective and less effective approaches are*?

Bobbie: So it is a community of the willing to some extent, people who are invested?

Steve: It is a community of the willing to some extent. We are a small organization. The products and guidance that we produce are created by the effort of members' employees who band together and work to share their experience and then deliver a common picture of what organizations ought to be thinking about doing as they tackle software security. We started out producing sort of heavyweight documents that took a long time to deliver. There are still some of those, very valuable, on the website. Fundamental Practices for Secure Development; dealing with third-party software, how to do threat modeling, are some of the recent ones.

But more recently we have focused on trying to be agile. Now we are trying to build smaller products that we deliver in the form of blog posts. We did a series of blog posts on building a security-champion program, for example, at the beginning of last year and like half a dozen blog posts over the course of a month to enable anybody who wanted to, to say, *OK, what does a security champion do, why do I want one, and how do I get started rolling that out in my company?*

Bobbie: Would I be wrong to connect this idea of a security champion to the role you had at Microsoft in trying to gain advocates to push a secure development lifecycle?

Steve: It is a little different. I was leading what we would call the Central Secure Software Team that defined the tools, set the process requirements, managed the training, what have you. Security champions are people in a product group who are focused on enabling that product group to deliver secure software, whatever that product group delivers, whether it's a Windows or a database-system or an Adobe Reader or what have you. There will be security champions in the product group – the engineering group who are the local go-to experts, local advocates, local interfaces to the broader security program. That is a very common model in industry, and it has worked out very well for a lot of companies.

Bobbie: So in a Microsoft context, those would be the people you would have worked with in the product lines that are there?

Steve: That is right. In Windows and SQL and Office and so on.

Bobbie: Can we talk for just a few minutes about what your catalysts were at Microsoft for trying to step into these security pushes?

Bobbie: Well, sure. Back in 2001 particularly there was just this series of security incidents, or maybe if you want to call them security flaming disasters, that is OK too. The Code Red worm, the Nimda worm, the UPnP vulnerability in Windows XP, which turned out not to be a big deal, but it got a lot of press. In parallel with that, customers were yelling at us and yelling at our executives. Actually, at one point, Gartner published an advisory that said, *Gee, if you are running the Windows web server, IIS, you might want to think about changing to another product*. So that is sort of an attention getter. And internally as well. People at Microsoft took a lot of pride in their work. They wanted to be an industry leader, a technology leader. So, all of those things came together and really sort of said, *We have to do something*.

Bobbie: Yes, I was on the operation side of Code Red and Nimda and others in an ops center responding. I was certainly personally particularly thrilled with the idea that you were trying to change some of the structure and culture in one of the big product lines that was out there. So how do you… It strikes me that you have always been an outcome-driven person. And I think about measurement in this space, right? How did you measure success in Microsoft as you were taking this on? And how do you think about that today?

Steve: There were a couple of things. You want the product to be more secure, but we don't know how to measure that directly. In fact, I was talking to a professional colleague, a pretty well-known vulnerability researcher, about some of our efforts over the years. This would probably be four, five, six years ago. He said, *You guys have done a good job, and I know you have done a good job, because the free-market price of a Windows zero-day vulnerability has gone way up from what it was 10 years ago*. That just to me is absolutely right, and it is a metric. It is an outcome-driven metric. It is not one we can cook or could cook. Short of that, how do you measure?

What we did was really two things. Number one, the requirements that we put in our secure development processes were really driven by experience of vulnerabilities. So, *How do people attack our systems successfully*? Those requirements are explicit enough, so that you can tell very clearly whether the product team has in fact met those requirements. If you look at the copy of Microsoft SDL version 5.2, that is a free download on the web, and if you look somewhere, I think that may be in the appendices [Roughly pages 112-114], but there are specific static-analysis tool bug numbers that are *must-fix*. That is measurable. It is cheap to measure. You run the tool and it tells you. It is cheap to measure whether you have gotten all of them—you just do a query in your bug-tracking system, and it will tell you. So that is meeting the requirements. Then the requirements to actually improving security, that is driven by experience with vulnerabilities and experience having to fix or not fix vulnerabilities going forward.

Bobbie: The interesting thing about those two examples, and I think one of our struggles in measurement, is that on one hand your requirements… The people doing the measurement have the ability to see the link between their action and their result. That outcome measure, that is measured by someone outside of the organization that… It is a really useful outcome measure, but bringing it back in and showing what actions complied, that is sort of the gap today.

Steve: You can say, *Well, there is some sort of an integer overflow vulnerability, somebody is going to write an attack and exploit it, and some terrible thing is going to happen*. *So if I had only just fixed that one, then things would have come out much better*. But in the security world, you just cannot attack it that way. I mean Michael Howard, who co-wrote the SDL book with me and one of the key developers of the SDL at the beginning, he used to say, *Just fix it*. It is not a matter of trying to second-guess whether somebody can exploit this. It is not a matter of trying to second-guess whether it is really a vulnerability or not, you know. If it violates the requirements, just fix it, and then you won't have to worry. Because probably, if you try to guess whether it is exploitable or not, you are going to guess wrong.

Bobbie: At some point the energy spent to that could be better applied to just the fixing it in that space. You have said that *Security assurance has been the most challenging topic I've had to contend with in my 45 years working on computer security*. Why, and how have you thought about where we are relative to methods and processes and formal validation?

Steve: Why is it challenging? Because it is really adversarial. I mean, it is really an emergent property of systems. You do not have a clear definition of when a system is secure in general. Back in the '80s, AI System Under the Orange Book, we tried doing mathematical models, and the mathematical models that were tractable enabled you to build a system that was not usable and nobody wanted to buy. So that was not very good. What we do instead is just to be driven by reality. I mean Rick Proto, who used to be the tech director for information assurance at NSA, used to say, *Theories of security come from theories of insecurity*. Start from how people attack systems and then take that class of attack and frustrate it. Take another class of attack and frustrate it. Take another class of attack and frustrate it. Somebody comes up with a new class of attack—you'd rather they didn't—that's OK, understand it. Frustrate it. Best case, your good guys, the red team that works for you, comes up with a class of attack, and you frustrate it and nobody ever knows it was there.

We actually had one fun case 2005, 2006, where some of our really brilliant security researchers internal to Microsoft came up with a class of attack that was really pretty bad, pretty pervasive. Then they built a tool that enabled us to find instances of that vulnerability. And we made that tool mandatory under the SDL. And starting with Windows Vista, we just wiped out all cases of it. And nobody ever knew. Maybe for its 20th anniversary, I will get them to publish a paper. But it doesn't matter because there is no system in support where that vulnerability still lives.

Bobbie: That is a beautiful, perfect case, isn't it?

Steve: That is beautiful, perfect case. That is what you would like, but you know that case has happened occasionally. Now, how do you replicate it?

Bobbie: What are the circumstances that enabled that to happen, and how do you create those? It is an interesting thing. I am struck by your language: *Find a category of attack and frustrate it,* compared to this idea of formalism. Frustrating is not as precise. Perhaps it is just you have made it… You put so much sand in the gears that you have moved beyond the adversarial, moved beyond compared to *What is the level of formalism that verifies that you are not going to have something?* Was that intentional?

Steve: Not that intentional, but it is probably not a bad model. One of the things we do is to try to remove vulnerabilities, instances of vulnerable code, when we find them. We know we are not going to do a perfect job of that. Some of the other things that we have focused on are what we call mitigations. Basically, an old example is, 15 years old now I guess, the non-executable stack-and-heap memory.

Bobbie: Right. I very much remember this.

Steve: The overruns, the errors are still there, but you cannot exploit them, or at least you cannot exploit them as easily. Microsoft and some of the SAFECode members have done a lot over the years with trying to follow that philosophy, not as an alternative to writing secure software but as a complement to writing secure software.

Bobbie: Right. The interesting thing to me about that example and the idea of it as a complement to secure software is the software gets you so far, and then at some point the platform has to pick up, and how you mitigate the gaps that are caused in that transition. I think that that example is a good one there.

Last year at USENIX 2019, you said that a big lesson you learned from the rollout of the SDL is that *secure software depends on*—I am going to get this right—*enabling secure development rather than measuring security compliance.* We keep trying to compliance-in security. How does your experience help us think about not taking that tack in the future? Is there some lesson from that that we should—

Steve: I think there very much is. And that really sort of goes back to the history of the Windows security push. Okay? At the time of the Windows security push, my team, that was sort of driving the security push, was like four or five people with 8,500 Windows developers.

Bobbie: The numbers did not —

Steve: We are not going to review all the code and find all the security bugs. Although I think that was a model at some point before I joined that team. What you can do and what you have to do is to motivate the developers to write secure software, train them on what that means, and then set them loose to doing it and provide tools, training, process, best practices, and advice when they need it, and that is a scalable approach. I have literally seen companies—and I think all SAFECode members are past this—I have literally seen companies where they have a security team—they say they have a secure development process—and have a security team that comes in after the software is done, and then they code-review it or they run their tools or they penetration-test it. Then they find bugs, and then they pass them back to the developers after the developers thought they were done. The developers hate that because they thought they were done. Managers hate that because they thought they were ready to ship. You wind up making “risk management decisions,” which too often mean, *Well, I can’t afford to fix this. I’m going to ship with it*. So, that is my model of what a compliance-driven software security program looks like: not so good.

In comparison, even the Windows security push, we trained people, we gave them tool outputs as they were doing development. They knew what bad code looked like. They knew how to replace it with good code. They came up with an edge case. They had somebody knowledgeable like a Michael Howard to turn to. At the end, we did not have a perfect product, but we were close enough so that we could ship something that was a lot better than what we started with or what the prior versions were. That is really my view on compliance. I mean you still can measure, but the measurement ought to be pretty easy because the requirements are pretty explicit. The tools and testing tell you very precisely what you should not have done. If the tools and testing do not find anything, then you have probably done a pretty good job.

Bobbie: One of the things that Dr. Michael McQuade, the vice president of research here, did in his work with the Defense Innovation Board is cause DoD and others to think about software in different categories—safety-critical software is perhaps different than not-safety-critical software—and apply different methodologies perhaps to those two. How do you think through that relative to this? It strikes me that while the requirements are going to be different in the end, the methodology might be somewhat the same.

Steve: For something security-critical, protecting the nation's sensitive secrets from the Internet, maybe you are willing to pay the price of doing the compliance thing and the post-testing. But for an awful lot of systems, most systems and most software that runs anywhere where people consider their information important or private or sensitive, you really have to do a base level of secure development. Because what we have found is that vulnerabilities in odd places can result in bad consequences. I think that one of the challenges is we would really like to see secure development practices be pervasive or universal or as close to that as we can get them.

Bobbie: How we write software has changed so much in the last… I know from when I learned eons ago. How do you think about what tooling or what is important for developers today in how software is developed today? Is there some —

Steve: One of the challenges when we created the SDL at Microsoft, we worried initially about C, C++, and C#, the three common languages internally. Today, many organizations that are doing development are dealing with 8, 10, a dozen, 20 languages. You have to do secure development in each of those. I talk a lot about scaling. That is a problem that really kind of poses a scaling problem. The more languages and frameworks you are dealing with, the more different secure development toolsets and standards and practices and training you are going to have to have. And if you are a small organization dealing with 20 or 30 languages or frameworks, that can get overwhelming, challenging. Then it is a problem, but there is not much of an alternative. At least we are starting to see the emergence of static-analysis tools, code-analysis tools, best-practices testing—things that individuals and organizations can pick up and learn and apply. But, I would say the diversity of languages and frameworks has not made our lives easy.

Bobbie: The cost of sustaining multiple code trees is really high. Then to think about sustaining multiple infrastructures to support all of that. Within that, is there a set of tooling that you would still like to see that is still necessary?

Steve: The things we talk about—the canonical SDL model, if there is such a thing—are threat modeling, which is really a design-analysis technique, static analysis, and more generally secure-coding practices, dynamic testing, fuzz testing if you are in the C, C++ world, but dynamic testing in general. Then the last I think is configuration testing, basically just to make sure that the files and objects that you lay down are protected properly in the environment or on the system. Those are sort of general. Then you kind of have to specialize then for environments. If it is Linux, it is different from Windows. If it is Python, it is different from C++.

Bobbie: So the interesting thing to me is this idea that that kind of infrastructure and tooling is a different model than that, which some people have, of *Anything can be written in anybody's basement.* So this idea of really thinking through your design analysis and infrastructure.

Steve: It can be affordable, but if you ignore that, that is a significant risk. One of the other things that has changed a lot today is the dependence on third parties, dependence on open source. Basically, everybody uses open source. Everybody uses third-party components. So, what is the assurance of secure development of this thing that you copied off some repository in GitHub? What is your standard for deciding whether to accept that or not? A SAFECode third-party components document that we released just a year or two ago talks a lot about best practices for third-party components. Ideally you would like your third-party components to be written to as good a secure-development process as you have yourself. It is not always the case, but that is what you would like to aim for.

Bobbie: Yeah. The supply chain has gotten much more complicated, both in terms of what you are getting, where you are getting it from, how much transparency is or isn't in that supply chain. I think about that a lot. We think a lot about that.

One of the things that I find really interesting is this push for speed. I think speed changes security a little bit in terms of how long something might be at risk. So, how do you think about what we are doing with Agile methodologies and DevSecOps and how that aligns with the work you are doing in software?

Steve: I had to face that with the MSN group at Microsoft, because they were… We had come out with this SDL thing, the Security Development Lifecycle, and they said, *Well, that is all very well if you are doing a Windows release that is going to ship in three years, but I am shipping this afternoon. It just doesn't work for me.* We had a program manager named Bryan Sullivan who knew a lot about Agile development, knew a lot about the SDL. He thought about it, and he enabled us to take the things that we said you had to do and then chop them up, parse them up. So there are some things, if you are shipping this afternoon, that is fine, but your code still has to run through your compiler tool chain before it gets deployed. So there are some security measures, some tools, that can just be integrated into that tool chain and their SDL requirements. So you just meet those as you go, every time you ship. There are some—if you are doing the broad architecture, the design of a system—that is something you are doing at the beginning of a project, and then it is only a major rev where you reconsider that. So, if you threat model, you do that once, and you have probably got a threat model that is good for a while, and it tells you what to do as you go. Finally, there are some kinds of testing that you have to run that take a while to do. Maybe you do those periodically rather than doing three days of testing every afternoon. So, you take a little risk before the next test cycle completes, but if the test cycles are ongoing, it is not a huge risk. Overall that process was how we tackled Agile security for the Agile development cycles. That worked out pretty well for us. I think probably that same model applies even for DevOps, DevSecOps, what have you. If you just think about parsing things out that way.

Bobbie: The interesting thing to me is the potentiality with all the tooling today to get data that can then be used, and you can push your risk management earlier in the process. You can push your security arguments earlier in the process. You can push your validations earlier in the process. And as you can speed up the release of hopefully ever-improving quality and assured software, you can change a bit of the dynamic.

Steve: If you can get the tools, and if you can integrate them into your developer desktop or development chain, then that is a great way to get the assurance up and the residual vulnerabilities down.

Bobbie: It would be interesting if we could get to a point where your third-party software has some data from its tooling about it that you can then incorporate. There is a little bit of a vision out there that we are all working towards.

Steve: That would be a great vision if we could realize it. I am sure a lot of organizations would be very happy to know that some coding best practices, that some static analysis, had been run on the third-party components they, we, are consuming.

Bobbie: Some of the software bill-of-materials activities, trying to see, at least knowing what is in there. Then the next step about could you put some details about what are the quality standards of the thing and others.the quality standards of the thing and others.

Steve: I have been a little bit of a skeptic on the software bill of materials. I think it is super monumentally important for developers to understand their bill of materials and what they are using. When people say, *End users need to understand the bill of materials*, I sort of ask why and what are they going to do with it. That is sort of a harder case.

Bobbie: For me the utility is in the architecture and the design and the implementation at the development stage. I hear your comment. So one last question. What do you know now that you would have liked to have understood much earlier?

Steve: Oh, gee. I guess what I know now that I would have liked to have understood much earlier is that getting to secure products, getting to software security, is really a matter of continuous improvement and root-cause analysis rather than trying to build the perfect model and then work for decades to implement it. I have tried the perfect model, and it didn't come out so well. There are people who still believe that that is what we ought to do, and in another 10 years it will all be nirvana, but I think if I can do continuous improvement for 10 years, things will be a lot better.

Bobbie: Thematically it is a culture-change question,I have heard you say. I think I also heard you say, *Hit the point of best impact for intervention, the developers*. On the software, *Think about good design patterns and threat modeling and other approaches to do it and constantly improve as you are going along*.

Steve: Constantly improve. Every vulnerability report: you take a vulnerability report, you do a root-cause analysis. Figure out, *Why did that happen? What can I do?* Not just to fix that vulnerability but to fix it and all its friends and relations. You just keep cycling that and over time you just turn the knob up and up and up.

Bobbie: And frustrate that situation.

So ultimately culture change is a big issue for security, and it strikes me that the work that you took on at Microsoft… You and I met I think the first time when you had just come to Microsoft and I was still in the DoD. How can we change this corporate culture to focus on product security? How did you go about that?

Steve: For us at Microsoft, the big turning points I think were two things. One of them was all the problems we had in 2001 that we talked about. The other was the trustworthy computing (TWC) commitment, the trustworthy computing email that Bill Gates sent. That was a combined project of Bill and Craig Mundie, who was one of our CTOs, and the late Howard Schmidt, the late, great Howard Schmidt, who was my predecessor at SAFECode and at the time had been the chief security officer at Microsoft.

The three of them, Howard and Craig really, encouraging Bill to create the TWC commitment and make that part of the company. That sent the message across Microsoft, and to the working-level developers, *This was something we were serious about*.

Another thing, sort of an interesting story about that. As part of starting this Windows security push and doing the focus on Windows Server 2003, we trained everybody. We had a room that held 1,000 people. We trained them in blocks of about 800 or 900, 10 times in 10 sessions over the course of a week, four-hour sessions. For each of those sessions, we had a corporate vice president, one of the intermediate-level managers in the Windows division, come in and introduce the training. If you have done corporate training, you know how that works: the corporate VP comes in, introduces the topic, tells you why it is important, and then leaves. In the case of the Windows security-push training, corporate VP came in, introduced the topic, told you why it was important, and then sat down to take the training. That is a message you do not see in companies very often. It tells you what is important and what is not. I think that worked for us pretty well too.

Bobbie: So, allies from lots of places really pushing forward and corporate leadership attention.

Steve: These people's VPs, these people's managers [are showing]: *This is important enough for me to spend four hours of my time learning it. Maybe it is important enough for you to spend four hours of your time really learning it.*

Bobbie: The interesting thing, another interesting nexus between you and I, I met Howard when he was on reserve duty working in the DoD CERT, which at the time I was running. So it is really a small world in those activities. Miss him a great deal.

Steve, thank you so much for being here today. It was really my pleasure to talk to you about your successes, your lessons, things that did not work out quite so well. Really the progress and focus on secure coding has been wonderful to see for us in the industry.

If you would like to learn more about SAFECode and what it is doing to identify and promote best practices for developing more secure and reliable software, hardware, and services, visit SAFECode.org. To learn more about the CERT Division and the work of our researchers at Software Engineering Institute, please visit our website. Thank you.