3 Key Elements for Designing Secure Systems

Featuring Tim Chick as Interviewed by Suzanne Miller

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.

Suzanne Miller: Hello, and welcome to the SEI Podcast Series. My name is Suzanne Miller, and I am a principal researcher in the SEI Software Solutions Division. Today, I am joined by my colleague and friend, Tim Chick, the technical manager of the Applied Systems Group in the SEI CERT Division, to discuss designing secure systems. Welcome, Tim.

Tim Chick: Thank you, Suz. Good to see you again.

Suzanne: You have been on our podcast series before, but it has been a little while. For those who do not know you, let's start by having you tell us a little bit about yourself, the work you do here at the SEI, and what the coolest part of your job is. That is my favorite question.

Tim: The coolest part is to work with wonderful people like you.

Suzanne: Good one.

Tim: Really, just being able to constantly learn, explore, and apply those things on real projects with real customer problems. I find that very rewarding. A little about myself, I have been with the SEI about 17-ish years. Most of my time here has been really focused on software engineering practices of various types and constructs. The last decade or so, I have been mainly focused on cybersecurity aspects of the software engineering lifecycle. Really focused on, *How do I build, design, and operate secure systems*, which is a slightly different perspective than maybe traditional cyber folks think about. Traditional cyber folks really focus on, *Here is my software, how do I build this moat and defend my castle*. Where my goal is, *Let's engineer a defensible and sustainable castle, so I do not need all those moats and all that other nonsense around it. It is secure by design.*

Suzanne: Exactly.

Tim: That has really been my focus, especially the last five, six years.

Suzanne: And *secure by design* is a theme across all of CERT at this point, and so this is right in the sweet spot of where you guys are working. That is what we are here to talk about today. And, as you mentioned, we are looking at building secure systems throughout the software development lifecycle, not just at the end. We are not bolting security on, as we used to say. You recently had a webcast, which we will link to in our transcript, and you talk about the scope of insecure system design, which I found very interesting, and the problems that can arise from insecure system design. So how big a problem are we talking about?

Tim: Yes. The problem is, we think about nine out of 10 breaches are due to defects in design and code. The average data breach in 2023 costs a company almost $4.5 million. This huge tangible dollar amount impacts our U.S. economy in trillions of lost productivity due to cybersecurity breaches and lost data and things of that nature. And, it also impacts us not just from a data perspective, but even physically. There are examples of people destroying physical property through access via the Internet. And so this whole cyber-physical aspect just compounds the issue because it is not just the Internet, it is actually people's safety can be affected by. . . And conveniences, the electric power grid, air conditioning in these hot summer days. All those things are now impacted by the Internet, right? Because it is an interconnected world.

Suzanne: Yes. And so secure design does not mean that we eliminate all

those problems, but it does mean that we eliminate, as you said, the largest problem, which is the defects that are in the code that lead to vulnerabilities and that sometimes lead directly to issues. What are some of the barriers? If this is such a problem, and it has such big effects, why do people not just build secure systems?

Tim: Because it is hard, right? I mean, it is engineering. And part of it comes to us as a software industry in general. While software has enabled amazing things, we are actually not very good at developing defect-free software. If you think about any time you open an application, or you do anything, you click those readme files. There are license agreements that probably most people hardly ever read. But if you actually read it, it actually says, *User beware. You use at your own risk. If this breaks something, causes harm anyway, it is your problem because you chose to use it.* That is basically what a user agreement says, and where part of that comes from is they are almost always defective. Quality is a major core issue, and the problem with that is we have done some studies, and 1 to 5 percent of every defect is actually a security vulnerability. It is something that could be exploited, and so it just compounds our poor quality. That said, there are things one can do to mitigate or reduce the risk of developing insecure systems. Start thinking about security as a design constraint and as a design element. Part of that is developers in general are very positive people, very confident in their abilities. What they do not think about naturally is how could someone misuse or abuse a feature that they are giving someone. Because the reality is, especially in Agile, the biggest thing is the value of customer. *The customer wants X, I am going to give them X*. Well, from a security perspective, you have to say, *Okay, the customer wants X, let me step back and go, 'How can I implement X so it can't be misused?'* That is the piece most people skip in their fundamental design and engineering thinking.

Suzanne: Yes. We have concepts like abuse cases, right? That is one of the ways that we try and highlight for developers that we need use cases of how is it intended to be used. But we also need abuse cases, which are ways that it could be misused or could be threatened by negative actors. One of the complications that is paramount in today's world is it is not just the software developer hand-coding everything that they are going to use in the system. Many of our developers use open source libraries, and then we have other elements of the supply chain. I am using somebody's router that has software in it. Sometimes those supply chains are three or four or six levels down. How do those aspects of modern software development complicate this problem?

Tim: Reality is, well over half of all software built today, really like 70, 80 percent, of most software is actually either third-party or open source software, right? From that perspective, an engineer needs to think about, *Just because I Googled something, and this library showed up as the most popular in my web search browser, does not necessarily mean it is the one I should use, or it is the most secure*. In open source, there really are some basic hygiene things you can do that are very easy to bring down at least that initial set of risk. One is, if I am doing open source, what is the number of maintainers? Does this open source library that I am considering have a vibrant contributor community or user community? Why is that important? Well, if there is only one person maintaining this thing, you have just sustainability risks that you are adopting because if something happens to that person, or they decide they are no longer interested in it, you no longer have this tool or library that is being sustained and patched and maintained. The other part is the number of days since the last submitted [contribution]. A lot of open source communities, you can actually see their process for contributing software. Is there a peer review process? Is it being evaluated and validated in some way? GitLab now, they provide a capability to run some basic static code analysis tools. You use their open source communities and some of their open source tools. There are things, some basic hygiene stuff, you can look at to make sure you are not making it too easy for your adversaries.

Suzanne: Yes.

Tim: The other one that can get you is, if you really have access to some intelligence and some other things, the people who are submitting code to these open source [projects], what are their usernames? What are their pseudonyms? Are those names associated with actual vulnerabilities and actual bad actors? Are they even hiding who they are a little bit? These really basic things are the one-on-one analysis one should do when considering an open source application to include in your product or your service.

Suzanne: One of the things I have heard people say is you should treat open source like COTS. When you bring commercial off-the-shelf software in, you test it, you check it out in your environment. You check things like you are talking about, make sure the vendor's valid, all those sorts of things. The vendor validity is something that is completely different in open source, but that idea of, *Do not bring something into your house until you have made sure it is safe*, is really the theme that you are talking about with both open source and other suppliers, right? That is the thing.

Tim: One key difference there is that for open source I would never take

binary into my application. I will take your open source, your source code. I will recompile it. I will put it in my own container if I am doing Docker containers. I am going to do all that myself. I might even take that source code and run it through static and dynamic testing tools to make sure there are no common weaknesses. If I have some access to look for like an SBOM and go through, because even open source sometimes uses other open source, right?

Suzanne: Right. Yes, yes, yes.

Tim: I might say, *Were there any known CVEs associated with this application?* If I am purchasing an application, yes, maybe I will accept the binary from a trusted vendor. But even then, I should be thinking about, and maybe even asking them, *What does their software development lifecycle look like? How are they ensuring security in this binary that they are providing me?* We are expecting them to do good engineering, just like I would expect my developers to do good, secure, thoughtful engineering practices. If you are a big enough vendor, a big enough procurer, those suppliers will answer those questions, either through documentation or onsite visit.

Suzanne: Right.

Tim: Yes, you should definitely treat them both the same. That one caveat is, I would never download or use a binary from the Internet. And people do it all the time, which just floors me.

Suzanne: Really?

Tim: Yes.

Suzanne: The developers I know just do what you are talking about. They bring in the source code, and a lot of times, modify it. You need to understand the open source well enough to know that your modifications are not going to create vulnerabilities. There are all these cascading effects.

Tim: I think the biggest one is people, out of convenience, will take Docker containers from unvetted sources, right? Because it is convenient, it is easy. Now there are places where you can go, like Iron Bank for DoD [Department of Defense], right? Okay, that is fine. But Iron Bank will tell you their vetting process, their security controls, how they are managing that. Okay, that is a trusted supplier of containers. But you'd be surprised, people still do it.

Suzanne: You mentioned SBOM, which is a software bill of materials . . .

Tim: Yes.

Suzanne: . . . and that is one of the newer concepts in some circles, so can you just explain a little bit about what is it that an SBOM gives you that helps you with building secure systems?

Tim: If a third-party gives you a binary, you cannot really just go through and look at what library calls or what other third-party software it is using to exist. You ask them for an SBOM, and there are some tools that will evaluate binaries and try to build some SBOM data, but the idea there is do I know what versions of what other aspects build this application? Because what I do not want is Log4j, for example. Most applications have logging capability. Most people do not actually build their own log capability, they get a library. I am going to say it wrong, but I think it is Log4j, they had a known vulnerability a while back, right? Well, I want to make sure that version of that library is not in the version of software that I am procuring, or I am adopting. SBOM should be able to give you that information of not only the name of application X, but that it is made up of these 50 other applications and third-party open source vendor software, and what version of those things it is. Because then I can take that information and I can look into any CVE repository and say, *Okay, are there any known vulnerabilities with version Y of this particular sub-component?* If there are, then my entire application has a vulnerability, even though there might not be a known CVE associated with this application that I am actually buying or using. That is the idea of an SBOM, it gives me the hygiene and the pedigree of the application, and everything is used to make it what it is. Did that help?

Suzanne: I know there are some viewers that appreciate that. We are big proponents of SBOMs because of exactly what you are talking about. And it is not just the security aspect, but it is also in terms of quality. There is open source that works as expected, and then there is open source that looks good on the surface and not so much underneath. You want to know what you are getting, right? You do not want to be buying a pig in a poke, so you want to know what it is you are getting to. A couple of barriers to secure design that we have talked about: one is the developer either knowingly or unknowingly bringing in open source that is dangerous in some way, adds risk to the software that they are building. Then another barrier that we have talked about is the idea of giving the pedigree and having all the information about all your suppliers that may be difficult to get. Not all open source libraries provide that kind of SBOM pedigree that you are looking for. What

are some other barriers? Are there any other sort of notable barriers to building secure solutions that you want to highlight today?

Tim: The other part is the actual engineering practices. I mentioned abuse cases, but there is this concept of doing threat modeling. If you are so large a system, you may be doing model-based systems engineering. If you do model-based systems engineering, you have design diagrams. You have basically a digital twin or design of what the system is that you are building. And you will see this a lot, even in software, for really large applications, especially if you are integrating your software with a big manufacturing thing like a boat or a plane. Because they are doing this stuff by default, and you are kind of fitting your software into their architectural construct.

Suzanne: Right.

Tim: But then you can step back and you can say, *Okay, what is a threat scenario for this element or this subcomponent of my system?* And you think about, what is the activity, what type of actors, because not every actor is someone outside your organization. You have insider threats. You also have just uneducated engineers who make mistakes. What action would they take? And then you also think about what the attack vector is. There is the MITRE CAPEC that you can look at, which gives you very different techniques that an adversary could use to exploit your design. The idea is if I develop some basic design scenario or threat scenarios, I begin evaluating my architecture against those things. The other ones, even more fundamental, are requirements. People like writing user requirements. People like writing system requirements, but they never stop and say, *Okay, what are my security requirements?* A lot of people go, *Oh, we will just be complying with this standard*. If it is the federal government, it is RMF, risk management framework. If it is in the banking sector or using credit cards, there are some other standards out there that they have to comply with. You get a lot of security stuff, but compliance is not a *shall* statement. There are lots of ways that I can be compliant with a standard, but not actually build a very resilient, secure system, right?

Suzanne: Yes.

Tim: That requires requirements of intent. I just state what I would expect, how would I evaluate that attribute of my system, which is more than checking a box for compliance. With all these developers, they build to requirements. They do not build to specs. If it is not a *shall* statement, yes, they will read the specs. They will try to make sure that is all impacted, but

they often grade themselves by their ability to meet the requirements. You take the time and actually write good security requirements that can propagate to different levels of abstraction, to different levels of your subsystems, because not every piece of software is responsible for every security aspect of your system. You need to take a holistic perspective to it. You need to engage as part of the engineering life cycle and engineering practices and stop trying to be this one-off. Come to the game and actually be a member of the engineering game and actually build something. That is what I am saying to all those security analysts and security professionals. It is that you have to engage with the developers. And a good bit of what I do is translate cybersecurity speak with engineering speak, because they come from very different backgrounds. Lots of them talk past each other. It is not intentional, it is just they have different vocabularies, and we need to translate those things.

Suzanne: Right, right. Good engineering practices, intentional security requirements, translating between the cybersecurity world and the engineering world, being, I will say, a savvy consumer of open source and your supply chain. Any other mitigations that you would say are kind of in your, I got one more for the top five, if you want the top five, that you would recommend?

Tim: You talk about secure by design, it is really becoming very popular and of interest to the global community. DHS [Department of Homeland Security], their saying is, *Secure by design, secure by default*. They have added a second element: secure by default. And what that is getting after is, as an engineer, when I build something, I deliver it to a user or an operational environment. Current practice in most applications, if you think of any application you might buy off the Internet or download, they make it really easy, so it works in almost any environment. But what they do a lot of is, they actually just turn off all the security features, so it just always works. A lot of government folks or banking or heavily regulated industries, they spend a lot of time and energy hardening an application on a server or a system before their users can use it. That secure by default means, *I am going to give it to you in a secure setting or secure configuration. Now, you as a user can decide to turn off some of those things, but it really flips the script. Instead of giving you an open [product] that you are responsible as a user to secure it, I am going to give you a secure application, and you as a user can decide to turn off the security features, but it is hardened by default*. As a developer, as an engineer, you should think about that user experience and how to not incentivize them to turn off those features because if it is hard to use, a developer or a user would just turn it off because it is getting in their way. You have got to think

about your mitigation. You got to think about your strategy for implementation to not incentivize that bad behavior of the people using your products and services.

Suzanne: This has been, since the '80s when I first got into this, the conflict between user experience, we called it human factors engineering then, but the user experience and quality attributes like security, they are known to be sort of conflicting quality attributes. Because the better the user experience, unless I am really smart about it, the lower the security. Which is what you are just highlighting and vice versa: that if I increase the security too much, it becomes a negative user experience, and I may lose my user or my customer. If it is easier to get around the security than to use it, that is a bad thing. This is arguing for me that we really need—and I do not know if this is part of the CERT strategy, I have not really talked to anyone about this—but it seems like we need to get more training and education and user experience for our cybersecurity people, so that they have a better understanding of what elements of their cybersecurity strategy may be impacting their user's willingness or ability to actually engage with the security practices that they are promoting. Does that make sense to you?

Tim: Yes. I mean, I think almost everyone has experienced the password requirements that are so cryptic and so hard, you have no choice but to write it down because you have no way of actually remembering this password that you created to meet all the rules on how to create a password. The reality is a password is not what is secure. It is not sufficient. That is why we have so much multi-factor authentication now. You log into your bank account, you have to verify with a text message they are sending your phone, right? That multi-factor is something you know, like your password, something you have your cell phone. These things, they have always been fighting with each other. You have got to work through it, and that is really where data comes in handy. If I start collecting the user's experience and even build into my systems and data collection in terms of how people are using my product, how they are using my application, I can very quickly see them turning off things or changing my settings. And the question is, *Well, why is everyone in my user community turning off this one setting? Do they realize how vulnerable they are making themselves?* Well, one, maybe they do and do not care because it is such a pain for them, or they just do not know. Then you have to educate, or you have to figure out what is incentivizing that behavior if they went out, and they looked for the way to turn this off. Because usually, that is not intuitive, how to turn off certain features. But I do it even with my iPhone or whatever. Sometimes I am like, *This is driving me crazy. Well, let me go Google how to turn off that latest upgraded version where*

they turn something on. There are the behaviors you look for. And if you are answering these types of questions in your website, then maybe you can see, *Well, why is that page the most visited page on my website*? You can also think about it from your other internal processes and documentation. You collect metrics that way too, to get some insight into what your users are reading and looking into, and what is on their minds.

Suzanne: This conversation highlights for me how rich an area secure system design is and not just design implementation operation. I do want to encourage our viewers to look more deeply into some of the things you have talked about because I know we have technical notes and blog posts on many of these individual subjects as well as sort of the overarching work that you and your team are doing in the overall space. Switching over, what are some of the transition resources that we have and that we can offer to our viewers to learn more about this. Do we have courses? I am sure we have websites. We have websites for everything.

Tim: We have lots of SEI blogs. That seems to be where my team likes to publish the most people read them and it is more readily available. You can do web searches, and they come up when you are interested in specific topics. The other thing we started to do, we have a Secure Software by Design event. Our second annual is coming up in August [2024]; it is free. The reason we did this is there are lots of security conferences out there, but I felt a lot of the security conferences are really focused on cyber operations, cyber defense activities, and not necessarily on the engineering practices used to build these applications. The real focus of our event is on the engineering aspects. *How do I build better products*? We have a rich agenda that is coming up. Like I said, it is first week in August, I believe. There are still maybe a few seats left before we can hit our capacity, for those who are interested. That is one way they could do it. We do have some courses. We have C, C++ coding standards practices. We have a virtual course on software assurance that Dr. Carol Woody designed. We do have those things as well. Lots of technical reports, things of that nature. We are really out there to just be a trusted advocate to tell people because we are not selling anything. That is the great thing about the SEI. We are here to educate and improve the state of the practice, in software engineering, because we are the Software Engineering Institute. re the Software Engineering Institute.

Suzanne: Okay. We actually will not get this podcast live before the event, but so for that secure by design event, I am assuming you will have some assets published after that, that people will be able to look at as well.

Tim: We try to balance it because there are lots of virtual [events], and we really just wanted an engaged audience. What we are doing is this year, we did not do it last year. Last year, we just had it purely in person. This year, it is still purely in-person, but we are going to record the presentations. We are hoping to release those recordings post-production.

Suzanne: And we will link to that event and its resources in the transcript. We will link to other things that you have talked about here, the courses and the blogs that relate to this. All that will be in our transcript for our viewers, so thank you. I want to thank you very much for having this conversation. I always have a wonderful conversation whenever you come online or even when we are just out in the world together. I really appreciate you sharing all this with our listeners. I do want to find out what is next for you because that is always intriguing to see what is coming, not just what is happening right now. What are you going to be working on next, Tim?

Tim: A couple of years ago, model-based systems engineering, things of that nature, we built this DevSecOps platform independent model. I think I might have talked about it.

Suzanne: Yes, you have.

Tim: Yes. I think me and Joe Yankel, we were both in that conversation? There are talks of maybe actually making it a real standard, the standards body.

Suzanne: You are going to get into the standards world, are you? Okay, good luck on that.

Tim: We will see. But, yes, that is currently a possibility. We will see how it goes.

Suzanne: Okay.

Tim: I would like to get back into that and maybe updating it, doing a new release, a new version of it, because the practices and the technology are evolving. There are more cybersecurity elements that we could integrate into it than we did with the first—well, actually, we have two releases now that we have gotten to so far. I would like to expand security practices, make it a little bit more embedded, especially thinking and giving more examples of threat scenarios and how that really works into that process.

Suzanne: You are going to make that model secure by default.

Tim: We will see. But—

Suzanne: I could not help myself. I am sorry.

Tim: But that is the idea, right? It is, *How do I get these secure practices in engineering*? And you are producing material and doing that to support other folks. They can use it in education and teaching, and universities because that is really where the industry needs to get to. We need to update the core discipline that the academia is teaching to include security of the next software engineer, computer science major.

Suzanne: Right now, that is critical because those are the practices that they come out of school with are the practices that they bring to industry. If those are out of date or going in the wrong direction based on how technology is evolving, that is going to impact the quality of the software they build.

Tim: Right. That is where we are trying to get.

Suzanne: All right, your mission, you have got your mission set, so I look forward to talking to you in the future about how that is going. I have had a little bit of standards work in my past and it can be rewarding. It can also be very frustrating because trying to get a whole room of people to agree on anything. . .

Tim: . . .Is painful.

Suzanne: . . .is challenging. And especially if what you are asking them to do is change their behavior, which, that is really what a lot of this comes down to, is changing the behavior of developers to doing more secure practices that take sometimes more time, and that may be counterintuitive to the way they have been taught to work.

Tim: Right, and I think we are at a good spot where we are getting there with executive offices now having security officers in the C-suite. . .

Suzanne: Yes, yes.

Tim: . . .and things of that nature. People are asking those questions. People are wanting more security or wanting, it is becoming mainstream, the impacts of poorly secured software. In fact, airlines just experienced one

recently, right?

Suzanne: I was there, I was there.

Tim: I was so glad I wasn't traveling. I was like, *Oh, it is a good week not to be traveling*.

Suzanne: Yes, yes. As soon as they talked about that it was a software issue, it is like, *Oh boy, here we go*.

Tim: Right. And that was just a quality issue. It was a bug from my understanding. It was not actually a cyber-attack or anything. And so again, quality matters.

Suzanne: Yes. Absolutely.

Tim: That is fundamental to our discipline, to our profession.

Suzanne: Yes, and I am glad that you and others like you are taking on that mission because it is very important for us all. And not everybody has the tenacity to kind of keep pursuing it. That is something that I admire about all of you that work in that area. I have said this before, every time I do one of these security podcasts, I have a couple of nights of lack of sleep because the scenario-based planning side of me just goes, *Ah, it can do this, and this could happen, and that could happen.* And it is like, *Ah*, but that is why we need you, because I cannot do this.

Tim: Because yes, sometimes ignorance is bliss, right? It helps you sleep well at night.

Suzanne: There are times, yes there are.

Tim: At the same time, it is like, *Ugh*.

Suzanne: Yes. All right. I do want to thank you so much for joining us today and talking about this work. To our listeners, thank you also for joining us today. We will include lots of links in our transcript. Tim talked about lots of different things, and so we will be giving you links to all of them and hope that you will use those. This podcast series is available in all the places you can find podcasts, Apple, SoundCloud, Spotify, and of course, my favorite, the SEI's YouTube channel. If you have any questions about any of this, please do not hesitate to email us at info@sei.cmu.edu. Thank you.

Thanks for joining us, this episode is available where you download podcasts. Including SoundCloud, TuneIn radio, and Apple Podcasts. It is also available on the SEI website at sei.cmu.edu/podcasts and the SEI's YouTube channel. This copyrighted work is made available through the Software Engineering Institute, a federally funded research and development center sponsored by the U.S. Department of Defense. For more information about the SEI and this work, please visit www.sei.cmu.edu. As always, if you have any questions, please do not hesitate to e-mail us at info@sei.cmu.edu. Thank you.