**EPISODE 313**

[INTRODUCTION]

**[0:00:00.6] ES:** Security vulnerabilities are an important concern in systems. When we specify that we want certain information hidden, for example, our phone number, or our date of birth, we expect this to be hidden throughout the system. However, this doesn't always happen due to human error in the code, because programmers have to write checks and filters across the program.

In this episode, Jean Yang, assistant professor at the Computer Science Department in Carnegie Mellon presents Jeeves, a language that allows programmers to specify security policies more intuitively making it harder to leak information that is meant to be protected. Jean explains how Jeeves was implemented and how it can be used.

We also talked about what it takes to bring research concepts from academia to the industry. At the end, we had a very interesting conversation on how to educate a broader audience on the importance of security. This episode also appeared on The Women in Tech Show, a weekly podcast where women in tech talk about technology. If five shows a week of Software Engineering is not enough for you, be sure to check it out.

[SPONSOR MESSAGE]

**[0:01:35.3] JM:** You are building a data-intensive application. Maybe it involves data visualization, a recommendation engine, or multiple data sources. These applications often require data warehousing, glue code, lots of iteration, and lots of frustration.

The Exaptive Studio is a rapid application development studio optimized for data projects. It minimizes the code required to build data-rich web applications and maximizes your time spent on your expertise. Go to exaptive.com/sedaily to get a free account today. The Exaptive Studio provides a visual environment for using back end, algorithmic, and front-end component.

Use the open source technologies you already use, but without having to modify the code, unless you want to, of course. Access a k-means clustering algorithm without knowing R, or use complex visualizations even if you don't know D3. Spend your energy on the part that you know well and less time on the other stuff. Build faster and create better.

Go to exaptive.com/sedaily for a free account. Thanks to Exaptive for being a new sponsor of Software Engineering Daily. It's a pleasure to have you onboard as a new sponsor.

[INTERVIEW]

**[0:03:05.9] ES:** Jean Yang is assistant professor at the Computer Science Department in Carnegie Mellon. Jean, welcome to Software Engineering Daily.

**[0:03:14.2] JY:** Thank you.

**[0:03:15.9] ES:** Security vulnerabilities are becoming more common. We are even getting used to seeing that thousands or even millions of accounts were compromised. In 2016, Yahoo said one billion user accounts were hacked, and we have also seen that it's possible to hack a car and take control of the driving while a person is driving in the middle of the highway.

Now that we are moving to a paradigm of more connected devices with the advent of Internet of Things, it's clear that there can be a lot of security risks. What is the state of the art in terms of security in academia?

**[0:04:06.2] JY:** There are many parts of academia with which we could talk about the state of the art. There are people who work on encryption, so what kinds of techniques can we use to protect individual pieces of data. There are people who work on the security of complex systems, so people who focus on, for instance, Internet of Things, all the ways Internet of Things can be hacked.

What I work on is programming languages and security. That's about how we can build our systems to be secure by construction. How we can improve our building materials so that we can decrease the chance that programmer sloppiness leads to vulnerabilities and how we can

make sense of the code so that it's easy for someone to come and inspect it and say, "Hey, this is the correct code." It's also easy for people to audit code after the fact. If there is a break in, they can say, "This is why it happened, and these are the parts of the programs that lead to the leak."

**[0:05:08.7] ES:** This is more about embedding security in a programming language, right?

**[0:05:13.7] JY:** Right. This is about thinking about security in programs from a first principle's point of view, because if we think about how security is done right now, a lot of things are after the fact. You build your help program and then you say, "Oh, wait. My toaster is connected to the internet now. This means that when I'm not home, people can turn on my toaster and set my house on fire, or they can make my toast saggy, or do all these other things."

Now, let's go back in and let's make sure that, for instance, you have to be on my home network to use my toaster, or these kinds of policies after the fact, or for Facebook. You share your location on your timeline and securities thought of as, "Okay. We have this whole Facebook that lets you search over people's data, that lets you go to other pages and then see connected data." If I'm say I'm at Disneyworld, and then I go to the Disneyworld page, it might say, "Your friend Jean happens to be at Disneyworld right now. Connect with her in some way."

To write the code for that right now, someone has to go in and everywhere that where a vulnerability might happen, they have to go in and write a check and say, "Okay. Jean's location shouldn't be shared at Disneyworld except with people who can see it." If you're doing Graph Search, "Oops! That's another place Jean's location can be seen," so we're going to have to protect that there. It's more that programmers have to think about security in a very manual explicit way every time, 'cause there's no one to handle this automatically for us right now.

Something that people have been thinking about in programming languages for a couple of decades now are; first of all, what would it look like to have data values be associated with the policies about when they can be seen? There has been decades of research on what is it mean to check that programmers are adhering to these policies? If I say my location can't be seen somewhere, what does it mean to automatically scour the entire code and make sure that my location isn't actually being seen when it's not supposed to be?

**[0:07:18.4] ES:** Most of the examples of these policies are about who can see what and when, right?

**[0:07:25.2] JY:** Right. Right. Right. Exactly. My research is about how can we make it easier for programmers to construct these programs in the first place, because even if you're able to check that a program isn't leaking information, someone has to write that program in the first place, meaning they have to write the right checks in the right places. They have to write the right functionality to say, "If Jean is seeing it, versus if her mother is seeing it, there's a different behavior that needs to happen." Someone still needs to go in and differentiate behavior in that way.

My work is about, "How can we build into programmers from the beginning? If these are your policies and this is what you want things to do, then automatically we can make sure that the programmer shows the right value based on the right viewers." As you said, it's about who can see which values. Also, in my work, what version of a value should be seen?

For instance, I might say, "My close friends can see my GPS location if they're near me." Otherwise, they see what city block I'm on, or maybe what city I'm in, or even what country depending on what I want them to see. All of these kinds of locations should be able to play well with the rest of the program, and that's something that I'm working on, getting a framework that can manage these kinds of multiple views for you automatically.

**[0:08:45.9] ES:** What you're talking about is important, because the way programmers have been doing this, there can be cases that can be missed. For example, in a presentation that you gave out, the Emerging Technologies Conference for The Enterprise at Philadelphia in 2016, you brought up an example of an vulnerability in Airbnb messages. Can you explain what this vulnerability was?

**[0:09:13.2] JY:** Right. With Airbnb, they have this thing where they want to be the mediators between you, and the guests, and the hosts. They will scrub instances where the host writes their phone number in a message, or something like that. They'll redact that information.

What I showed during this talk was there are some cases where Airbnb forgets to redact the information. I think in this case, it was in the e-mail preview of a message. It showed the host that, “Oh, dear guest, the place is ready for check in, and here’s my phone number in case you want to call me.” On the website, Airbnb scrubbed that phone number information so the guest couldn’t see it. In the e-mail, you could see the phone number and contact the host directly.

In this case, nothing bad is happening. It’s a leak that doesn’t hurt anybody, because the host wanted to give the information to the guest. It’s showing that Airbnb didn’t intend for this to happen, because they are pretty careful at scrubbing the phone number in most places. In this case, they forgot. You can imagine how in a similar way Airbnb, or Facebook, or one of these sites that has a lot of your critical data can forget to scrub some information that ends up harming the user.

**[0:10:31.1] ES:** Are these cases missed because the way that policies are enforced is through if/else statements?

**[0:10:41.1] JY:** Exactly. Exactly. Yes. The way the policies are enforced is the programmer has to write an if and else  conditional statement at the place where the data is being used, like, “Oh, if this string is looking like a phone number, then we have to redact it,” or they can maybe factor the code very nicely to do a library API call at that point. Everywhere that data might be leaked, the programmer has to think, “Oh, I need to do something at this point.” If they forget to do that, then information can be leaked.

**[0:11:11.3] ES:** The way that you explained earlier to embed this in a programming language, is it at the minute where you declare a variable, like in addition to the type you specify?

**[0:11:22.4] JY:** Right. Right. Exactly. How we’re thinking about it now is when you’re declaring a piece of data in addition to declaring the type, you can declare it to be sensitive or not, and then you can declare permissions about how it’s meant to be seen. This is based on these many years of language based security work that I’m talking about that looked at how do we label sensitive data, and there’s also been a lot of work on what do the policies look like. How do we talk about when data can be seen? Who it can be shown to and how we can relax the policies based on extenuating circumstances or something like that.

 

**[0:11:57.3] ES:** To first illustrate all these concepts in programming languages, you develop Jeeves. Is Jeeves a programming language or a library for an existing programming language?

**[0:12:11.2] JY:** That's a very good question. We call Jeeves a programming language and research. What we mean in research by a programming language, it has a well-defined semantics. The one way I think of programming languages is as applied logic. In programming languages research, how we think of a language, it's a logical system that we can apply and write a real programming system out of. We've proven a set of properties about this system, and it does certain things that we want.

In programming languages research, abstractly, a language is just like a close system. We can reason about with us that have properties that we understand. How these languages can get implemented is either standalone languages that people build compilers or interpreters for, or as libraries that they graphed on to other languages.

How we have implemented Jeeves is as what's called an embedded domain specific language, an embedded domain specific library. We've implemented it as libraries in both Scala and Python, and they are a little strange as libraries, because they actually rewrite the program as they're running it. That's where the language's library really comes in. It's not just like this is a set of calls. You can make — And that's our language. Actually, when you include the Jeeves library and you say, "This is a Jeeves function, we actually go and rewrite your program on the FLY.

And so that whenever you're doing a conditional statement or you're making a variable assignment, we're making sure that everything happens according to the policies you stated by calling our special conditional statement that does the appropriate policy bookkeeping instead of the regular one, and our assignment function that updates our state based on the policies and where the values are coming from and that kind of thing.

**[0:14:03.0] ES:** Are there any disadvantages to rewriting code on the FLY versus if the policy implementation was added to Python, or Scala?

**[0:14:15.4] JY:** Oh, year. Rewriting code on the FLY is very expensive, because you're calling over the AST as you're going, and there's definitely a nontrivial runtime overhead of doing that. In fact, our latest work has been looking at if you can state policy specifications as fancy type declarations, and then how we can use a compiler to in-search hacks into your program for you by analyzing the code before you run it. That gets rid of your runtime overheads.

Unfortunately, it's hard to do that right now for regular languages. We're doing it for a language called LiquidHaskell, which is Haskell plus even fancier types. One direction of future research is to look at if a language had less fancy types, or even no static types at all, how we can adapt these techniques.

**[0:15:07.4] ES:** However, this technique of a library for Python, or Scala, is good enough to prove the research idea, the concept, right?

**[0:15:18.1] JY:** Right. Right. Exactly. There are ways to make it more efficient. We can look at, "Okay. In these cases, we don't actually have to rewrite the program, because we know these program properties." Yeah, to show that the language actually works and to show, "Hey, this is how you can play nicely with an existing language that people actually use, that lots of people actually use, "that this is a good first way to go.

**[0:15:40.6] ES:** What are the different labels of policies that can be specified in Jeeves?

**[0:15:46.7] JY:** The  policies are actually very expressive. Anything you can write as a program, you can express in Jeeves. You can have — For instance, you can have simple policies like you have to be logged into the system to see this, or you have to be part of this list. Even you have to be part of this list, starts being less simple, because that's a policy that can include a database query right there.

Then, you can say, "You have to be a member of the list that's protected to see the protective list," or, "you have to be within 50 feet of the secret location to see the location." These are even fancier policies that might even depend on the value that they themselves protect.

The really neat think, I think, about the Jeeves approach is that because the policies are managed by the system and because we really tried hard to support very expressive policies, your policies can just be arbitrary database queries, arbitrary code, and we'll make sure that the policies are resolved in a way that doesn't leak information.

[SPONSOR BREAK]

**[0:16:59.2] JM:** Indeed Prime flips the typical model of job search and makes it easy to apply to multiple jobs and get multiple offers. Indeed Prime simplifies your job search and helps you land that ideal software engineering position. Candidates get immediate exposure to the best tech companies with just one simple application to Indeed Prime.

Companies on Indeed Prime's exclusive platform will message candidates with salary and equity upfront. If you're an engineer, you just get messaged by these companies and the average software developer gets five employer contacts and an average salary offer of $125,000. If you're an average software developer on this platform, you will get five contacts and that average salary offer of $125,000.

Indeed Prime is a 100% free for candidates. There are no strings attached, and you get a signing bonus when you're hired. You get $2,000 to say thanks for using Indeed Prime, but if you are a Software Engineering Daily listener, you can sign up with indeed.com/sedaily, you can go to that URL, and you will get $5,000 instead. If you go to indeed.com/sedaily, it would support Software Engineering Daily and you would be able to be eligible for that $5,000 bonus instead of the normal $2,000 bonus on Indeed Prime.

Thanks to Indeed Prime for being a new sponsor of Software Engineering Daily and for representing a new way to get hired as an engineer and have a little more leverage, a little more optionality, and a little more ease of use.

[INTERVIEW CONTINUED]

**[0:18:47.8] ES:** One of the early experiments that you did with Jeeves with a real world application was building a conference management system for a small conference at MIT. What were some of the examples of the policies that needed to be in place for a system like this?

**[0:19:09.8] JY:** Yeah. For an academic conference, this is the example that researchers like to use. One reason we like to joke is because an academic conference management system is something that someone reviewing a paper has to have used in order to get to the paper review. It's also something with very complicated policies potentially, because you have these different roles, so you could be a paper author, or you could be a paper reviewer, or you could be the chair of the program committee in charge of managing all the reviews.

Depending on your role and what stage of the conference it is, you can see different things. If people submit a paper, a paper has a title, it has authors, it has a body. The authors are something that's very sensitive potentially. There's in-reviewing. There's something called blinding. Most conferences, how they choose to run it in our field is there's either single blind, which is the reviewers can see the author identities, but the authors can't see the reviewer identities, or there's double blind, which is neither the authors nor the reviewers can see who each other are.

There are policies about when authors and reviewers can see each other's identities. There are also policies about when reviewers can see other people's reviews. You can imagine, if one person reviews a paper favorably, or unfavorably, it can affect other people's opinions. Many of these conferences have a policy that says that while you can't see other people's reviews until you've submitted your reviews, and maybe you can't even see who the other reviewers are until some stage in the conference to preserve some kind of notion of fairness.

You have policies about who can see reviewer's identities, and which reviews can be seen. Also, authors might not be able to see the reviews until a certain date where they've discussed all the reviews and now they're allowed to be released, et cetera. These are all the sorts of policies that go into a conference management system.

**[0:21:04.4] ES:** What was it like using Jeeves who established these policies for this web application?

**[0:21:11.0] JY:** It was fairly straightforward using Jeeves. The nice thing that we showed was if you use Jeeves, you can write the policies once instead of as checks across the entire program, and so you just write it once. Then, when you're doing search — There are some examples of when — Of leaks that have happened in these conference management systems. Many of them include the search interface.

If you go to your paper directly, you can't see — If you're the author. You can't see necessarily, "This reviewer said this," or, "I think they're going to accept the paper," or something like that. If you go the search and you say, "I want to search for all the papers that have been accepted." I think that's actually an actual bug that's happened before. You can use that if your paper has been accepted, or you go to your search and then you say, "I want to sort the papers by score," and maybe you can't see the score of your paper directly, but somehow the sorting function could have seen your paper, your paper score, and then you can infer the relative scoring of your paper based on that.

All of that, you don't really have to worry about if you use Jeeves.Because we were actively developing Jeeves at the time, the bigger problem for us was actually getting everything to work and run out of memory and things like that, because we were doing this very expensive dynamic rewriting of our program and exploring all possible program pads and all these stuff. For the policies, it was fine. It was more — When you're building a research language and you're saying, "Okay. We'll just do the most expensive thing for now to show that our language works conceptually." Sometimes you can't do that if you're trying to do something for real.

A big thing we realized was, "Well, the semantics of the language are great, because you have these guarantees of correctness." In practice, you really don't want to be exploring all your programs pads, and you want to find ways to decrease that and that kind of thing. That was where we ended up spending most of our time.

**[0:23:06.8] ES:** What about the database for this application?

**[0:23:10.8] JY:** Yeah, that's an excellent question. In building the conference management system for the first time, I had a big realization which was, "Oh shoot! Up until now we've been

thinking about security and privacy only in the runtime layer, essentially, because if we're doing things at the "language level" we're really only thinking about a small part of a web application. There's also the front-end. There's also the entire database. What I realized then was the database is a huge gaping hole. You put your policies alongside your data.

Really, where your data is being stored is the database. If you want to use the database for any queries at all, those are not managed at all by Jeeves. That's what motivated a few years of work on the Jacqueline Web Framework in which we extended the computational model of Jeeves to a SQL database, and we looked at — This is how you enforce policies across the application and database, and this is how you can get this idea to work for real.

**[0:24:13.0] ES:** This is great, because by working on a real world application, even if it was small, you found downsides for the current implementation such as performance and thinking about the pipeline from a web application perspective. This makes me think about what it takes to take a concept from research and academia to the industry and the barriers for industry adoption. Why is it so slow to incorporate research into the industry?

**[0:24:53.1] JY:** I think there are a few reasons why it's so slow. One thing is that even our work on Jacqueline which involved extending the programming model to the database, it took a really longtime for people to see that it was useful, because, I think, for a while we didn't know exactly how to articulate this as an interesting research result even though it was. Two; people kept saying, "You did Jeeves already, how is this different?"

Part of it was just we needed to formulate the problem in a way that was interesting to people, but I think that academia also does not necessarily reward work on taking your idea and building a system out of it, because it's easy to say, "Oh! This is just engineering." Yes, a lot of work is potentially just engineering. It takes longer, potentially, to figure out what's interesting about it.

This engineering work often gives you really good insights about how to make things that are actually useful. It's definitely not something that's incentivized by academia, but it's much easier for people to judge the novelty of something that's completely novel than it is to judge the novelty of something that seems halfway between research and engineering, especially

because if someone hasn't gone and written a web application, they have no clue what it means to interact with a database.

For me, I really hadn't built a lot of web apps before that. I never thought about the database interaction. If you think about these are people who are trained to think about certain kinds of problems and they're not going out and building things necessarily, it's really hard to convince them, "Hey, if you go out and build things, these are the holes."

I think that the incentives are just not necessarily there. Even Jacqueline, what we built with the database is still very much a research prototype. I think that there are a lot of questions that would need to be answered before it's production ready. Some of the questions we're answering now; how do we get rid of the runtime overheads? How do we do all these other stuff?

It takes a lot of scheming to do stuff that is taking us towards something useful while being aligned with the incentives of academia, producing ideas that are novel, showing that, "Hey, this is another step towards building knowledge instead of just building a system." I think there is this whole area between what makes a very splashy good research paper and what actually is production ready, 'cause I think that the way to think about academia is we're de-risking ideas for industry. Even once we've de-risked the big things, there are still many questions to be answered that are smaller risks that aren't making money right away, that you can't build a startup based on, that it's not clear who's going to answer these questions.

I think that industry research labs have been in a good position to answer these questions, so Microsoft research, Samsung research, Ball Labs, and these kinds of places. They have some incentive to eventually push ideas into production, but they also have a lot of trained PhDs, trained researchers who think very academically. These places have been pushing things a lot further towards industry. I think outside of that, it's very hard to have the right set of incentives to do very risky work that is highly technical, very specialized, and might not work.

**[0:28:14.4] ES:** Do you think academia should change to a more balanced model where it does incentivize industry related applications? For example, the fact that Jeeves was working on a web application?

**[0:28:28.0] JY:** I do think that it would be nice if people realized that we do have these blind spots in academia, but there is a danger. If we move too much towards an applied model that there's a real blurring between what's academia, versus what's industry.

A problem is that it's just hard to tell what's a good idea from what's a bad idea, and there is something very nice and very useful about the fact that academia allows people to build on ideas that are not immediately useful, and such this term basic research that you might have heard before. It's talking about research that how do we get a jellyfish to glow and not like how do we cure this very specific disease that this many people have at this very moment?

Getting jellyfish to glow turned out — I think that wasn't the question, but how do we get things to glow based on the jellyfish protein? I don't know if you've heard about this, but that's like really changed neuroscience research and changed how people are curing disease and all these stuff. It started with some very basic questions that seemed irrelevant to a lot of people.

It's really good to have a space for that kind of work. I think it'd rather have academia on the side of being less practical, than more practical, because there are pockets of computer science that are very close to industry, and I worry about those pockets, because people are competing on, "How fast is this? How much money it can potentially make?" We're not industry. We're never going to be competitive with industry when it comes to those very concrete benchmarks. I think that if we get into a state where we're just competing with industry, we're going to lose every time, and we'll lose what makes academia so nice for insulating these kinds of ideas.

**[0:30:19.4] ES:** We're thinking more outside the box, because if it's focused on the industry, you would — Maybe it would have been shut down, the idea about glowing things, because it would have been, "What are you going to do with that?"

**[0:30:33.2] JY:** Right. Exactly. I think that when we first started working on Jeeves, we said, "Well, we want these semantics, but the only way we know how to do it is in a very expensive way." If we were an industry, people would have just laughed us out of the room and said, "No. You're never going to do this."

Around the time I started working on Jeeves, the summer before, I had internet Google, and they were rewriting their front-ends of something from Java back to C++, because they couldn't afford the memory overheads of Java. I would talk to them about programming languages and they'd say, "That nice and all, but we just can't afford the memory overheads." That's Java, not anything more expensive.

If that's the level of innovation that you're tied to if you want something production ready, you really can't get very far thinking that way. There is a lot of value in having a space where people say, "Oh! Just give us the newest most crazy idea and we'll accept it for its novelty."

**[0:31:26.6] ES:** Research from academia can be slow to be incorporated in the industry. However, eventually, it makes it there. One of the things you mentioned is programming languages, for example, like Swift have taken features that were incubated in research decades ago. One of the ways that you mentioned it might be faster to get it into the industry could be having startups adopt certain technology first. Why do you think startups can be a good fit?

**[0:32:04.5] JY:** One thing that I've thought about a lot is why big companies are slow to adopt certain kinds of new technologies, and a big reason I think is because they have really large legacy code bases that are tying them down in certain ways. If you're Facebook, you have millions of lines of existing code, and an existing code base.

If you're thinking about adopting a new language, or even if you're thinking about a new kind of programming tool, a big question is, "How does this play well with these millions of lines of code that I've already invested in that we've trained people to understand?" et cetera.

If you're in the business of making programming language and tools, one route to go is you make your code play with legacy systems which is one route I'm taking very seriously and I'm interested in taking with my research. The other route is if you want to build new stuff, you look at people who don't have that kind of baggage. I'm not saying baggage in a negative way, it's just something that people have to consider when they're making decisions.

I think that there's just less risk involved if you have no history of technology you've used before that anything has to play well with, you can take on these new ideas and you can move faster.

[SPONSOR BREAK]

**[0:33:28.5] JM:** When you are continuously deploying software, you need to know how your code changes affect user traffic around the world. Apica System helps companies with their end-user experience, focusing on availability and performance. Test, monitor, and optimize your applications with Apica System. With Apica Zebra Tester, Apica Load Test, and Apica Synthetic, you can ensure that your apps an APIs work for all your users at any time around the world.

Apica Zebra Tester provides local load testing for individuals, small teams, and enterprise DevOps teams to get started quickly and scale load testing as your needs evolve. Apica Load Test ensures that your app can serve traffic even under high load. Apica Synthetic sends traffic to your website and your API endpoints from more than 80 different countries, ensuring wide coverage.

Right now, you can go to softwareengineeringdaily.com/apica for a webinar about the real ROI of API testing. You can also find past webinars. Just how to optimize websites for fast load time. Go to softwareengineeringdaily.com/apica to find the latest webinars on load testing and lots of other topics, and check out Apica System for testing, monitoring, and optimization.

Thanks again to Apica for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

**[0:34:58.9] ES:** At the ET Conference, another thing that you mentioned is that we rarely see security focus startups. It might be because there are no flashing demos, or users are not educated yet about the importance of this. I think that with this idea, you confounded the cyber security factory, which is an accelerator for security startups. Can you talk of a little bit about its mission and what it is?

**[0:35:30.9] JY:** This came out of a conversation with a fellow PhD student at MIT, Frank Weng. We were talking about how the startup culture right now is such that you need certain kinds of demos and a certain kind of flash and slickness to get people's attention, and security really is

not that. We wanted to create a space for these kinds of highly technical solutions that required a very niche audience to understand the significance of. Also, we wanted to create a space for security in particular, because how you demonstrate a security concept is good is by showing that you've done the math and you've thought about things deeply so that you can guarantee the absence of certain kinds of behaviors, and it's very different than showing the presence of certain kinds of features, which is what this demo culture is all about.

We also felt that while there are a lot of certain kinds of security startups, so if you do like penetration testing, or bug finding, that's much easier to start in many ways, because it's about demonstrating the presence of something, "Hey, I looked through your codebase and I found these vulnerabilities." You can show something based on that. Even cryptography is getting a little more popular, because you can show, "This is how fast they can encrypt something."

There are many parts of security that are about building your system better, building your system in a more robust way that are more subtle. We wanted to create a space for thinking about how companies like that could communicate about their technology. Even for people who are working on ideas like that to realize that they could be a company, because if you're not seeing other companies like that and you aren't necessarily thinking in an entrepreneurial way, you might not think that this is an option that you have.

**[0:37:23.5] ES:** One of the key findings that surprised me from the two companies that initially I'm boarded with cyber security factors, was that they said, "Networking can really drive innovation." Why do you think that is?

**[0:37:41.3] JY:** It's something that we alluded to earlier in our conversation, which is that there are these gaps between how different people see things. Academics have one world view that we live in, and people in industry have another worldview that they live in. Often, the problems that academics think are problems, are not quite the same as what people in industry think are problems, and people in different pockets of industry even have different ways they look at problems and how they prioritize which problems are the most important.

The more you get people talking to each other, the more cross-pollination you have of these different views, and maybe academics can adjust our world view. We're still working on early

crazy out there problems, but it's directed in a way that is potentially more useful to people, more useful sooner, or just more useful in terms of the direction it's taking. There is a point where talking to people too much makes academics too applied, or makes industry people have strange views.

**[0:38:42.1] ES:** They cyber security factory have connections with potential clients for the participants of the program?

**[0:38:49.8] JY:** Yes. Frank has gone on to run it last year, and he's running it again this year, but through the years we've formed a lot of connections with potential companies who are interested in piloting these kinds of security solutions and/or incorporating these companies as part of them.

We talked to a company — I'm not sure how much of this is disclosable, but there is a company that was really interested in taking these teams and running pilots of their stuff as part of their own programs, and that kind of thing. Having a framework for doing this is definitely helpful for creating social, or some kind of corporate infrastructure for these companies to plug in.

**[0:39:33.7] ES:** Last question; in your opinion, what are some of the things that we could do to educate people more about the importance of security in software? Because I think part of the big problem is that a lot of people don't know about this, because they don't know what happens behind the scenes and the code.

**[0:39:55.2] JY:** Yeah. I read an article yesterday that talked about how better user interface design can help with informing people about their different security and privacy options and can get them to think more about how their security actually matters, because I think, right now, there's a really strange relationship that people have with their security in privacy. They're really not in control of it, because if you think about it, you download some app, let's say, Uber, or something like that, it says, "Uber wants to use your location." You can click yes and let Uber use your location, or you can click no, and then Uber doesn't work anymore. Something like that.

I recently had this experience with using WeChat, that said, “WeChat wants to use your location.” Then every time I wanted to start a conversation where I needed to search for a friend, it said, “It needs to use your location.” For the first week or two, I just said, “No,” and I would just wait for people to WeChat me, because I said, “No. I don’t want them to use my location.” I don’t know what they’re going to do with my location. It got kinda old after a while and not to be able to initiate chats with people.

Yeah, I think that, right now, one reason people might not care is they feel like even if they care, there’s not much they can do about it. These dialogue boxes really aren’t initiating dialogues or kind of coercing us to go along with certain policies about how our data is being used.

I think there are some different parts of this. One is that there needs to be more conversation about how users can have more control over their data. I’m not entirely sure how this can go, because, right now, the companies really have full control over what they allow, and so Uber would need to invest in this whole other path. If I don’t want to share my location with Uber sometimes, they are the ones responsible for making other functionalities possible, where they see my location only some of the time.

One of my hopes with my work is to address this, if there’s an easy way for me to tell Uber, “Okay. This is what my location data should look like to you.” Sometimes I just don’t show you my location and you have to deal with it. That starts more of a dialogue with Uber about what the policies should be.

The other thing is maybe we need to wait for once there are more capabilities for doing this. User interface designers can also do a better job of saying, “These are your options. You don’t just have to click yes right now. This is very coercive.”

I don’t think it’s the fault of the user interface designers. I think this is just what they have to do, but once we get to a place where can have this more of a back and forth between users and the creators of software about our data, then user interface designers can explore some more interesting user interface design options for like facilitating this dialogue.

**[0:42:43.0] ES:** Definitely. It shouldn’t all or nothing.

**[0:42:45.5] JY:** Yeah, ‘cause right now — Sure, you can say people don’t care and it’s our fault, but what are we supposed to do? Not use software? I don’t think it’s entirely fair to blame people.

**[0:42:57.5] ES:** Jean, thank you for coming on the show. It was great talking to you about security and programming languages.

**[0:43:05.6] JY:** Yeah, thank you so much for having me. This is fun.

[END OF INTERVIEW]

**[0:43:12.3] JM:** Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. Thanks again Symphono.

[END]