Kelly Lum, also known as Aloria, has worked in computer security for 14 years, and is currently a security architect at FlatIron Health, a healthcare technology company that focuses on oncology. (At the time of this interview, she was working as a security engineer at Tumblr.) Lum regularly speaks about reverse engineering at various conferences, including Black Hat, SummerCon, and COUNTERMEASURE. She also teaches as an adjunct professor of application security at New York University (NYU).

Did you start out with an interest in security engineering, or did that come later?

I wasn’t a very well-behaved person when it came to computers back in high school. So, when I started college, I went for general computer science. I always loved programming and messing around with things. Around my junior year, the school introduced a full scholarship in exchange for concentrating in computer security for two years, and then you worked for the government.

I jokingly said, “Well, you can only really become a computer security person if you hack into the FBI, and then they hire you instead of sending you to jail.” Which obviously wasn’t what I believed, but I didn’t realize that this was actually an industry and people did it for a living.

This was back in the early 2000s, when security was very young. I started out doing research on various network devices and eventually figured out that my interests were more on the application security side—especially since I have a background of loving to code and build out applications.

Many people were frustrated with the SFS [CyberCorps’ Scholarship For Service] program because the government service part didn’t focus enough on security engineering and overfocused on operations. Reasonably good software security people would get stuck managing a bunch of firewalls, which really bothered them. I’m hoping they’ve fixed that.

I lucked out and ended up working for the US Army. My job was less configuring firewalls and more operational security. It was like, “Here’s a whole bunch of intrusion detection systems and firewalls and network antivirus. We want you to audit them for security. Try to find vulnerabilities and write proof-of-concept exploits for them.” For several reasons, if you’re going to buy a firewall and put it into an Army network, you want to make sure that it doesn’t have back doors from whatever country it was produced in.

I have a trick question. What’s the difference between application security and software security?

People think about both types of security in two ways: as a software platform or as a specific application, wherever that might be. It might be on a desktop. It might be running on a server. It might be embedded in a piece of hardware. People tend to think, “I’m looking at code. I’m looking for specific exploits and vulnerabilities.” But it’s not just code or application analysis. That’s one part of it, and it’s one way to see how well the application or system...
has been engineered. But it comes down to: What are you trying to do? Are you just trying to find exploits that you can use to prove something, or sell to a bug bounty, or use for nefarious purposes—or are you actually trying to build out a piece of secure software?

When you talk about application security, it tends to be because you came to security through the OSI [Open System Interconnection] stack. You wandered up to layer seven and said, “Layer seven is called the ‘application’ layer. Let’s call this ‘application security.’” But if you’re a software person you’re like, “It’s software! You thought it was a car or a thermostat, but it’s actually software.”

I tend to look at it from the latter viewpoint, especially because I came from software and I tend to think of applications more in terms of what the developer or designer was thinking. But there are applications running on your firewalls. I became quite familiar with this during my first job out of college.

How much of what you do all day as a security engineer at Tumblr is taught in school?
Teaching application security at NYU, I’ve realized that everyone wants to learn how to be a hacker, and how to find vulnerabilities and write exploits. But you also need to understand how to securely design it and how to come up with the right requirements.

It has to be a mix—especially when you’re working for a company that’s producing an application. It’s producing a web app, it’s producing a mobile app, it’s producing an API. It’s not just sitting down and auditing code and penetration testing all day. It’s sitting down with developers, looking through architecture plans, doing architecture reviews. Now we have requests for comment, where we talk about the way we want to design things, and it’s a conversation rather than us lecturing the engineers in a meeting.

Instead of getting people to focus on the soft stuff, such as secure development lifecycles, mix that in with the things that students tend to think computer security is and use that as a carrot to get them to look at the other stuff as well.

This isn’t just for Tumblr, but for other organizations where developers think of things in terms of the software development lifecycle. Get them in a room and have a capture-the-flag exercise so that they say, “Hey, there’s a ‘cool’ component of this, too.” It’s not this mystifying, ephemeral thing where you’re either born with it or not.

Hands-on stuff like that is essential because if you look at, say, what the government does in the name of software security, it often amounts to a bunch of bureaucratic box-checking.

I’ve been in other roles where it tends to lean more toward that than more technical work. And I feel like those jobs are necessary, especially if you’re working in an industry where you have regulations to comply with.

But I also feel that you have to change, break out of your routine every once in a while, and look at things from a different perspective. That way, you don’t get one part of your brain stuck in a rut. It would be like if you always work out your arms and forget leg day.

Can you compare and contrast working for the government and working in the financial services sector, from the perspective of geeks like us?
I don’t want to say they were completely different worlds—just because I was in a very niche part of the government. Other parts might be more closely aligned. But it was a completely new experience from private industry. Things move extremely fast. You’re making money. If you get it out two hours late, it could drastically impact your revenue or the product’s reputation.

It’s also a world where you have to make a lot more justifications for what you’re spending money on. You can’t say, “Hey, we’re going to spend $3 million on some new firewalls.”

You really have to sell yourself as: “We’re not making you money like your core business is. But we’re in the business of making it so that you don’t lose money in the long term.” I hate paying homeowner’s insurance every month. But if my condo ever burns down, I’m going to be a lot happier that I have it.

Let’s move on to another topic. I think code review technology has come a long way since 2003. What’s your view on that?
If you have the more automated stuff nailed down and are giving developers useful information, then it’s great because they don’t have to wait for somebody to manually
check every little pull request they’re merging. But the key words there are “giving them useful information.” You can’t just say, “Oh, we got a lot of money on it.” It’s: “Here are 20 things you need to look at.”

One of the big problems in the industry, especially when it comes to application security, is that we tend to overfocus on finding bugs without telling anybody how to fix them. “We found something, but this is your problem. You have to take care of it.” This is why many developers really, really despise security.

Because you’re making fun of their ugly baby.

That’s one of the refreshing things about Tumblr. When you find a security problem, you either go to the developer and say, “This is an issue, and here’s how to fix it,” or, “Let me help you fix it.” Rather than, “I found a problem. Assign a ticket. I don’t have to think about it anymore until you triage.”

What kind of things should code review tools do that they don’t today? This includes, say, reviewing dynamic languages like JavaScript and Ruby.

We build our own tools at Tumblr, which kind of leads into the answer to your question. With commercial tools, you try to fine-tune them based on the available rule sets.

When you go out to buy a car, some car dealers say, “You have model A, model B, and model C. Maybe you can get heaters on the seat of this one.” Or, if you go to a different car dealership they’re like, “You can get these wheels, and this type of transmission. You can get five-speed or six-speed. You can get automatic.”

I want more modular things. I want this widget and this widget, but I don’t need these others.

Integrated development environments [IDEs] have a new, lighter-weight set of tech. But you can’t do dataflow analysis properly inside the IDE, so you get more false positives. So, you only look for certain kinds of errors that way. Also, you have to look for other errors at build. I think the tech is coming along, and I think everyone should use it. But I do think it’s a problem if you don’t do code review at all.

I’d like to see more correlation between, “Let’s look at what sort of errors are spewing from our logs. Let’s run some of our JavaScript through a headless browser or....”

I wouldn’t do it for a nuclear power plant that way. That’s one of the problems. Automation is great but SecDevOps seems to overlook design review completely. Again, you have to focus some attention on design while you’re building. I’ve made this distinction for years between bugs and flaws. What do we do about flaws because we can’t really automate that yet—we don’t know how.

It’s one thing to scan some code and say, “You’re not sanitizing something.” It’s another thing to say, “This thing has been architected so that your authentication, in the right conditions, has a bypass.”

Let’s switch gears once more. In your talks, you’ve described data loss prevention [DLP] tech as keeping honest people honest. Is that good or bad?

It depends on which type of person you’re trying to keep honest. If you’re trying to help somebody who deals with a lot of data but might not be the most technically proficient, or might not understand the difference between trust levels, you’re preventing somebody from copying and pasting something from a classified document into an email. I’m sure many people work with data that isn’t supposed to be in certain places—for instance, they email it to themselves so they can work on it on their home computer. Or, they print it out so they can read it on the train home and then accidentally leave it on the train seat.

That sounds like herding sheep. But you can’t really herd wolves. So, if you’re buying DLP to herd sheep, good. And if you’re buying it to herd wolves, don’t buy it.

I would say if you have sheep and wolves, don’t expect it to protect against wolves. And sometimes you don’t know the wolf is there until it’s too late.

How much security tech do you think, in operations land, relies on that “Well, we’re just going to make the sheep do the right thing.” philosophy?

I’m always resistant to say things like this but, I feel that a lot of times these solutions, especially DLP, have their place. There are situations in which they’re good at doing whatever they’re supposed to do. But I also think that there’s a misconception, and this might not be the fault of the buyer—that the seller is exaggerating its capacity or its ability to solve problems.

We’re still in a phase in security where we have a lot of very intelligent, talented people. But it’s not across the board. We have people who are making purchasing decisions or who might be in a less mature organization, and they don’t have the tools or the information they need to make educated choices about certain things.

I have a question that’s a little difficult. Your honesty as a successful functioning human who is battling mental illness is absolutely inspiring.

I’m wondering whether you think there are any benefits to your own diagnosis that are unique and useful in the field.

I don’t want to perpetuate the stereotype of the brilliant or struggling artist with mental illness. And I wouldn’t want to romanticize it.
But I do think that having this problem for a very long time has forced me to be very analytical. Because when you’re having a thought or you’re in a mood, in order to survive you have to analyze: “Why am I feeling this way? Why am I thinking this thought? Is this thought true? Is this a realistic thing that I’m thinking?”

I’ve had to be analytical and investigatory about my own thinking processes, and I think this has made it a little less difficult for me to do that same thing for computer security. You have to analyze things. You have to ask, “Why was this done? What was the thought process behind this? Is this normal? Is there an alternate way of approaching this problem?” I do think it’s made it easier for me to use that internal process on external things.

Your honesty and openness will be hugely helpful to a lot of people, so kudos for that.
At first I was frustrated because there didn’t seem to be many other people talking about it. So I said, “Well, why don’t I just talk about it?” And I really have—it’s almost selfish, but I do get a lot of people asking me for advice and thanking me, so I really benefited from it.

There’s a word for it: leadership. One last question. You have some really great travel pics on your blog. What’s your favorite part of Helsinki? I really like everything about that area, not just Helsinki. I also took some time to go over to Estonia, and loved everything about the old architecture versus the new architecture. The people were so nice. The biggest highlight of my trip was New Year’s Eve. I was worried about getting a table for dinner because I didn’t make a reservation. I was just wandering around, and I saw this little place that looked cozy—a very traditional Finnish restaurant. I went in and said, “Feel free to kick me out, but is there a tiny little corner that you can put me in?” And they’re like, “Actually, yeah. There’s this one corner with only one seat and a table, and we never get anybody to sit there because people always come in large groups.” I had a wonderful reindeer fillet and then tried bear for the first time. I’m always about trying the weird meats.

The Silver Bullet Podcast with Gary McGraw is cosponsored by Cigital (part of Synopsys) and this magazine and is syndicated by SearchSecurity.

Gary McGraw is vice president of security technology at Synopsys. He’s the author of Software Security: Building Security In (Addison-Wesley 2006) and eight other books. McGraw received a BA in philosophy from the University of Virginia and a dual PhD in computer science and cognitive science from Indiana University. Contact him via garymcgraw.com.