Editor: Gary McGraw, gem@cigital.com

Silver Bullet Talks with Matthew Green

Gary McGraw | Cigital

Hear the full podcast at www.computer.org/silverbullet. Show links, notes, and an online discussion can be found at www.cigital.com/silverbullet.

Matthew Green, an assistant research professor at the Johns Hopkins Information Security Institute, talks about the difference between theoretical and applied cryptography, blogs, and back doors.

You've spent a lot of time straddling the gap between academia and the corporate world. Can you explain the difference between theoretical and applied cryptography?

It turns out, and I was surprised by this, that a lot of people who do applied crypto, meaning they write software and do things with cryptography in the real world, don't seem to like theory very much—and vice versa! Academic cryptography researchers don't spend a lot of time up to their elbows in code, and that gap has started to become a problem for the software world.

If you're in academia, learning about crypto and getting excited about it, how do you make the transition to applied crypto?

I would advise that you look at real systems—there are tons of them and tons of code as well, especially in the open source area. We use major protocols for all kinds of things, but nobody has poked at them academically. There's a wealth of stuff that you can explore and get published, so start doing it.

Your blog made a splash recently when you got a takedown request from a misguided dean. Tell us about the controversy first, and then the actual content.

There's not much to tell. I was contacted by a reporter from *ProPublica* who had some background questions about encryption for me. I thought, "Wow, this guy seems convinced that the NSA is spying on us all. I hope he doesn't write an article saying that I'm the one who's saying that." I tried to be very careful, and then one day, I look in *The New York Times*, and there's this huge article about how the NSA is spying on us all and specifically how they're breaking our cryptography by inserting bad standards and putting back doors into products. It was all well sourced and amazing, so I wrote a blog post fleshing out what was in there, making clear I'd never seen any classified documents—you can read it at blog.cryptographyengineering.com. The post was just to put some meat on the bones and state that we didn't know too much yet. I linked to *The New York Times* article, grabbed an image off *The Guardian*, and that was it.

I had this post up for about a week, and then I got an email from my dean saying, "You need to take this down right away. There's been a report that you're hosting classified material and using the NSA logo improperly." So I wrote to a colleague, Avi Rubin, and said, "There's no way I'm taking this down, but what are we going to do?"

When it hit Twitter, I remember thinking, "Uh oh, here it comes."

I didn't know it was going to turn out the way it did. I assumed that a few nerds would get upset about it, but a bunch of other people did, too, and it became a huge deal.

It became a huge deal because we have First Amendment rights. This

About Matthew Green

Matthew Green is an assistant research professor at the Johns Hopkins Information Security Institute. His research includes techniques for privacy-enhancing information storage, anonymous payment systems, and bilinear map-based crypto. Green has a PhD in computer science from Johns Hopkins.

notion of a chilling effect is especially disturbing and concerning when it gets all the way to academia.

I spent the day thinking I was going to lose my job, which is liberating, because I got to have lunch with my grad students and be pretty relaxed about it, but it was a scary time, and it has made me think twice about some of the things I do and say now.

Let's talk about the content. According to the leaks published by *Ars Technica*, *The New York Times*, and others, the NSA has tampered with standards, weakened protocols, bugged Skype, broken crypto keys, infiltrated telecom companies, and decrypted SSL connections. What are your thoughts about these activities as a practicing cryptographer?

It's like being a doctor and finding out that somebody has been sneaking into your blood supply at night and tampering with it. It's that bad. Maybe I'm being too dramatic, but that's how I feel. We're out there, trying to build secure systems, and it's really, really, really hard to do. Meanwhile, you add to that almost impossible equation a person actively trying to sabotage you. We can't trust anything. Maybe that's okay—living in a world where the NSA can spy on us—and maybe the NSA isn't the adversary most people care about, but it's still upsetting, because we don't know what's good anymore. We don't know if it's just the NSA or if their weaknesses are exploitable by other people.

In the blog post, you state that "software is a disaster." Do you think we're getting any better at software security?

We're getting better at software security, in part thanks to people like you, but unfortunately, I don't think that kind of improvement translates equally to all areas. At financial institutions, people are willing to spend money on security, so yes, they're getting better software. The problem is that there's a lot of software out there that we still depend on, such as OpenSSL, Apache, and the Linux kernel, that aren't getting as much of that kind of attention right now.

There was this notion that many eyeballs could magically solve that problem, but it's naïve.

I don't think there are that many eyeballs with the kind of skill it takes to find these subtle bugs.

And economics dictates that those who do have such eyeballs tend to be expensive.

They're expensive, and apparently, as we learned recently, some of them are actively working for the other side. How do you fight that? I don't know.

Here's the $2.1 billion question, then: What should companies and individuals do about this?

We're in the phase where the goal is to diagnose the disease and figure out how bad it is. Adding back doors to commercial software, breaking standard stuff—I think that crossed a line. The government has plenty of perfectly good reasons for intercepting communications. I'm willing to accept that we need an NSA, but we should draw the line at breaking security that could make us less secure overall. Our first task as academics and as security people is to figure out where the problems are. A big lesson on that front is RSA, a division of EMC, being included as one of the companies accused of adding a back-door random number generator [RNG] to its BSafe cryptography toolkit.

What's the impact on applied cryptography? We tell people not to roll their own crypto all the time, and here we are, putting disastrous chemical waste directly into a standard.

The random number generator was proposed by NIST. It was supposed to be the secure, government-approved RNG, but we now have every reason to believe it has a back door in it so that people can figure out random numbers. There's no reason in the universe to put this thing into a product as a default. I've been over it a thousand times, trying to determine why anyone would make that decision. It's slow. It eats system performance. It's biased. It produces bad random numbers. And it's been known for a number of years to have a back door.

What's the difference between something that's a good random number generator from a Monte Carlo perspective versus a good random number generator from a cryptographic perspective?

If you're doing statistical simulations, you need random-looking numbers, and there are algorithms for generating them. But when you're talking about cryptography, you need something stronger. You need to know that no adversary is going to be able to predict the next random number, so we use different tools, one of them being a pseudo-random number generator with cryptographic components. What's different about this one is that it's

based on elliptic curve cryptography, which was all the rage back then but makes it fairly easy to hide a back door. You could have a public key in the RNG and somebody else can know the secret key—by using that secret key, they can actually predict the next bits, which goes against the way random number generators are supposed to work. It's disastrous for crypto.

If you use RSA BSafe in your products, and you've linked in that bad random number generator, it's probably time to go fix your stuff.
It's the default, so if you used BSafe and you didn't go out of your way to make it not the one you're using, then yes, it's time to go replace your stuff.

So what should companies do about this? The individuals and corporations who are counting on crypto to be done properly by the people they're buying it from—should we tell them, "Roll your own"?
That's a terrible idea. We should try to get people using standard open source. The idea that you could sneak a back door into a widely used, commercial, closed source library—I mean, it can only happen in a closed source library. It wasn't snuck in. It was published that it was there, but nobody paid much attention to that. We have to have more open source, more widely studied libraries, but even that's not the ultimate answer, because you can do things to open source software as well. And yes, it's just awful, messy code. Awful. Awful.

But we all use it.
We all use it because it's the best thing out there, which is sad—and exactly my point. Over the past few years, there's been a bit of negligence, partly from the research community and partly from industry, about replacing these not very well supported tools and doing things right. We just said, "Okay, it works well enough. Let's patch it and keep moving." I think we need a big investment now in looking at all our standards and all our code and seeing what we need to redo.

The NSA is supposed to spy, but we also somehow put it in charge of defense. It's like we didn't learn anything from the Cold War; everything from a policy perspective got ignored when it came to cyber. What are we supposed to do about that?
We have NIST, and its job is to produce the standards that we use for crypto. But it's mandated by law to work with the NSA, which has a lot of the brain and computer power that's supposed to be deployed. In fact, the NSA's mission is to secure our computing systems—part of it is, anyway—but NIST doesn't have the resources to look at SHA or AES all by itself; it needs help, and it's required to have that help. The problem is that we just don't know, nor will we ever know, if the NSA is definitely working in our defense and not against us. I don't know a solution to that problem.

But from a policy perspective, we're asking the NSA to work at cross purposes: "Please go out and spy, but shoot off your left foot when it comes to spying."
Exactly. You'd think that somebody at the NSA would have said, "Hey, this could come back and hurt us all." Somebody didn't make that calculation or count on having a Snowden, but there was, and now we're dealing with the consequences.

How do we manage to give the government some sort of a clue about security engineering? Most media coverage seems to be about network security.
Bottom line, we don't know how to build secure systems. A few people have started to make progress in it and train other people in how to do it, but until we come up with an entirely new approach to building software, we'll never solve the problem.

Did you figure out a way to teach this? Do you think you can teach it in school now that you're a research professor?
What we've done is kind of like trying to mop up the ocean—we work with good companies and make good progress, but for every company we work with, there are 20 or 100 that we don't. You guys have a more scalable approach: you're trying to train people to write the code, and you're writing tools to help people write code. I think that's definitely a start. But I also think that ultimately we're reducing the number of bugs, not eliminating them. Until we figure out how to write software that can't

be exploited—and, unfortunately, I don't know how to do that—we're going to be at risk.

How is academia creating people who can poke holes, find bugs, and patch them?
I saw a lot of promising work in building programs that couldn't be exploited. We all pushed for it, but I think we've realized that it's just too hard of a problem, and more recent work has turned to simpler challenges, such as trying to detect exploits. I think that's still promising work, but it's kind of depressing to see nobody out there coming up with the future of how to make software secure. It just doesn't exist.

We spend our time in computer science theory trying to build universal Turing machines and talking about what's computable and what's not. But other machines way down the Chomsky language hierarchy could do a lot less by their very mathematical nature. Why not figure out what we can do with those things to avoid security problems?
I agree that trying to bite off more specific problems that we can actually make some progress on is a big, big part of this. But people who try that approach then run into the wall of, "well, nobody will use it." There are people out there building these strongly typed, verifiable programming languages. Do you want to write software in this language? I don't.

It's more fun to have the power of assembly language on steroids, which is C.
Exactly. Everybody knows it, so you can get people to write for cheap. That's the bottom line.

I have one last totally off-the-wall question that I asked Avi Rubin many years ago: What's your favorite breakfast cereal?
I've always loved the Captain Crunch with the crunch berries. It's terrible for you, but I love it.

You're still allowed to eat that?
Not at all. I don't even let my kids go near it.

The Silver Bullet Podcast with Gary McGraw is cosponsored by Cigital and this magazine and is syndicated by SearchSecurity. ■

Gary McGraw is Cigital's chief technology officer. He's the author of *Software Security: Building Security In* (Addison-Wesley 2006) and eight other books. McGraw has a BA in philosophy from the University of Virginia and a dual PhD in computer science and cognitive science from Indiana University. Contact him at gem@cigital.com.