

SOFTWARE
ASSURANCEINTERVIEW:
SOFTWARE
SECURITY IN
THE REAL WORLD

Ann E.K. Sobel, *Miami University*
Gary McGraw, *Cigital*

In an interview conducted by *Computer* editorial board member Ann E.K. Sobel, Cigital CTO Gary McGraw discusses the state of software security and the BSIMM—a data-driven research project describing and measuring what successful organizations are doing to ensure software security.

SOBEL: As Chief Technology Officer at Cigital (www.cigital.com), Gary McGraw counsels businesses on building and running software security initiatives.

Gary, you have made many contributions in the area of software assurance in a relatively short period of time, including books such as *Building Secure Software: How to Avoid Security Problems the Right Way* and *Software Security: Building Security In*. From your perspective, what accomplishments are you most proud of?

McGRAW: What I'm the most proud of is something that I'm still working on. You know how it is when you're in the middle of a project—it's very exciting and fun to do. Although I'm proud of the books that I've written, the discipline of software security and, maybe more widely, software assurance, has matured significantly in the past 15 years. *Building Secure Software* was written in 1999, and *Software Security* was written in 2006. More recently, we've been studying actual software security initiatives and then reporting our findings.

We're studying real companies with real initiatives that have hundreds of employees working on software security every day. The BSIMM (Building Security In Maturity Model) model that we've built is available as a free download (<http://bsimm2.com>). BSIMM is a study of how 30 firms approach software security. (The "BSIMM Firms" sidebar provides an alphabetical list of 20 of the 30 firms that will allow us to talk about them.) Many of these companies are household names. It's likely that most readers will run at least some software from one of these companies because their software pretty much pervades the planet. Some of these companies have been focused on software security for more than a decade, and studying what they're actually doing is what I've been working on for the past year or so.

We launched the BSIMM project in March 2009. The first study included nine companies; since that time, the study has tripled in size. It's a data-driven model, and we've been gathering more data and adjusting the model accordingly.

Based on our observations of what these successful companies are doing, we can now state with some confidence that we know what actually works in software security and what people should be doing.

SOBEL: One of the surprising facts that I took away from my review of your project is that the majority of the companies you've been studying deal with financial services. I was surprised to see that only four of the companies work in defense. Are you surprised by that distribution?

→ BSIMM FIRMS

The BSIMM (Building Security In Maturity Model) project takes advantage of common ground between diverse methodologies. BSIMM is a descriptive study of the software security activities practiced by 30 firms. As such, the project is much less about what an organization *should* do for software security and much more about what leading firms are actually *doing*. BSIMM is free, distributed under the creative commons license, and available for download at <http://bsimm2.com>.

The BSIMM is an observation-based, descriptive model directly describing the collective software security activities of 30 software security initiatives. Among the 30 firms studied in the BSIMM project, the following 20 have graciously allowed their names to be used.

- Adobe
- AON
- Bank of America
- Capital One
- The Depository Trust & Clearing Corporation (DTCC)
- EMC
- Google
- Intel
- Intuit
- Microsoft
- Nokia
- Qualcomm
- Sallie Mae
- Standard Life
- SWIFT
- Symantec
- Telecom Italia
- Thomson Reuters
- VMware

McGRAW: Not really. It's probably more a reflection of who I know and the kind of work I do than it might be of the market. Cigital is a software security services provider. We have about 140 employees, and we provide services mostly to the financial services industry. Among the 30 companies in BSIMM, 12 are financial services organizations, including: Bank of America, Capital One, and Wells Fargo. Seven are independent software vendors (ISVs), including Microsoft, Google, EMC, and Symantec.

With populations so theoretically disparate as financial services on the one hand and ISVs on the other, you might think that it wouldn't be possible to build a model that describes the activities undertaken by both. But it turns out that it is indeed possible. In fact, now that we have 30 companies in our study, we have a statistically significant data set. We can do some straightforward statistical analysis like a T-squared comparison of activities between verticals. Not only can we say that there's an overlap in activities, we can demonstrate how overwhelming that overlap is. Among 109 activities, these two verticals have 100 activities in common.

It turns out that when you look at the data for ISVs and financial services firms, they're doing the same thing. A pithy way of putting this, as my coauthor Brian Chess likes to say, is, "There are no special snowflakes." The question is whether or not that insight also applies to defense contractors. The answer is that we haven't done enough studies of people working in defense to be able to determine what normal is. But in my experience,

the government has tended to be far behind the commercial market when it comes to software security, and even software assurance work. That may be because the overly bureaucratic, almost too rigid approaches to accreditation and certification like the common criteria are frankly just untenable. They don't work in a real economy in my view.

If you look at what the market has done to address the need for software security without engendering a huge bureaucracy, you see the sorts of activities that we described in the BSIMM. The activities have much more to do with actually thinking about and testing software and training people to do a better job than they have been doing without filling out reams of paperwork and getting a special stamp from the secretary of underwater basket weaving.

SOBEL: Given that you're finding this common core, do you think that the day might come when you step back and assess whether these are good activities or bad activities, somehow passing judgment in terms of what should be considered best practices or not?

McGRAW: The answer to that question is somewhat complicated.

As it turns out, we observed 109 activities. However, if there had been 212 activities, we would have reported that many. There was no constraint on either a minimum or maximum. We gathered the data first and used it to build the model. Of those 109 activities that we observed among the 30 firms, there are 15 that the most firms do. In other words, 20 or more firms, 66 percent, are doing these particular 15 activities.

That means there is a high density core among the activities that we observed. On the other hand, what we cannot say with any confidence is how much you should invest in one activity over the other 108. In addition, we have no way of directly measuring the efficiency or effectiveness of an individual activity.

We're building a community of like-minded people who are doing this professionally—people like Steve Lipner at Microsoft, Eric Baize at EMC, and Brad Arkin at Adobe. These people have a desire to get real data as well. They really want to know how they should invest, and they want the data to help them make decisions about adjusting the knobs on, say, the 12 practices that we have. (Each practice has multiple activities.)

We are collectively very interested in trying to determine whether we can get a handle on efficiency and effectiveness. As a community, we have decided that that's the direction in which we're going to take the research.

Ultimately, our research is going to evolve in two ways. We're going to continue to add firms to the population. I would definitely like to include defense contractors, the military, and government agencies in the data set. We're

also going to expand the work to try to get a handle on efficiency and effectiveness.

It's a significant challenge, and accomplishing it isn't going to be trivial, but we've already begun to meet that challenge to a certain extent.

We'd all love to be able to put some software in a magic box on which either a red light would light up, in which case we would throw the software away, or a green light would light up, and we then could be certain that the software is secure enough for use. But that's a pipe dream. Sadly, my belief as a computer scientist and as a software practitioner is that we can't do a direct first-order measurement of software security.

Instead, we have to retreat—and I use that word very carefully and consciously—to looking only at activities and creating what you might call a second-order metric. Looking at activities is not really looking at the thing you're producing. But our belief is that those activities do in fact lead to better software.

I wish that we could measure software security directly, but we can't. Instead, we have this activity-driven second-order measurement, and we build our software security measuring stick based on observed activities.

SOBEL: Unlike the personal software maturity model, where we actually assess our processes and the steps that we take in construction, it's kind of a removed activity, if you will.

McGRAW: That's right. The only difference between my philosophy and, say, Watts Humphrey's philosophy, who began his work long ago, is that in order to get a handle on security, I think we have to talk explicitly about security. We have to realize that attackers do have what philosophers would call an intentional stance. Thinking about the attacker's intent and thinking like a bad guy doing the wrong thing on purpose and intentionally trying to break a system makes security ever so much more difficult and interesting than something like reliability. I think Watts would argue, on the other hand, that reliability is a superset of security—if you just get your requirements right, security will come out in the wash. I just don't believe that.

SOBEL: Do you think there's any merit in either certification or standards as a way for companies to address software security assurance?

McGRAW: I do actually. Let me put a slightly finer point on it. I believe that the BSIMM has the ability to become a de facto standard, and in fact that's already beginning to happen. As the community involved in the BSIMM project grows, we're seeing a lot of cross-pollination, learning from each other, and a little bit of competing with the Joneses. This is great because the results benefit the common good.

On the other hand, if we approach software measurement only as a bureaucratic exercise in record keeping like the common criteria has tried to do, the results can quickly become obviated and pretty much useless.

There are some reasons for that which may not apply more widely to all certification and accreditation schemes but do apply, say, to the common criteria. One of the challenges is that the common criteria are wide ranging. Everybody said they wanted to assess all sorts of different products from firewalls to VoIP phones to laptops to coffee mugs, whatever is lying around. That means we have to create special protection profiles for each one of these targets of evaluation. Immediately that opens Pandora's box, and you can't ever get those evils back into it.

The problem is that every individual company with an individual product said, "We have a special protection



We have to realize that attackers do have what philosophers would call an intentional stance.

profile just for us," and soon the common criteria weren't very common at all. We weren't measuring anything in common—all we were doing was ensuring that somebody had to spend half a million dollars pushing paper around in circles.

My hope is that the BSIMM doesn't devolve to that sort of thing. The "no special snowflake" result that I described before is very encouraging along those lines. Let me give you a real-world example. The guys in New York who run financial services firms (who are all the butt of the media coverage these days due to their shenanigans with risk) had always asked the independent software vendors to provide some evidence that their code was secure. The software companies would by and large say, "Oh, yes, yes, I know that you make software, but we make software differently. We're special professionals and your software is also special, and these are not the droids you're looking for." They would try to pull some sort of Jedi mind maneuver.


But our results show that, in fact, you can ask the software providers to give you some evidence. Because, for example, the static analysis tools and results that they're getting are very similar to the static analysis tools and results that the financial services firms are already demanding from their own developers. There is no special snowflake! This is good, because it turns the lights on in the marketplace and allows us to compare measurements directly. The BSIMM work allows this measurement, which you can then use for various things.

Another concept that we've been thinking about (but we've never actually seen this in practice) is to imagine that

a firm has its own BSIMM score and knows to some extent (metaphorically) how “tall” it is. The company can make a statement to its subcontractors and vendors and say, “Look, we’re this tall, and if you want to ride the ride, you have to be this tall too.” Thus, the company could demand some sort of a measurement that is a proxy for whether or not its software is actually secure.

I hope that sort of thing happens. I do believe that sometimes accreditation and certification schemes and measurement can be used properly. The trick is how to go about actually having them be meaningful. We’re trying to approach this as scientifically as possible with the BSIMM.

SOBEL: It sounds like your view is that such activities would probably evolve from businesses putting forth different methodologies and tool support for measuring such things. Generally, certification activities tend to come from professional organizations or standards could possibly come from government agencies, thus a business model would be appropriate.



There are no value judgments in the BSIMM. Instead, we focus on observation.

McGRAW: Yes, that’s probably a reflection of the fact that I’m a business guy, and my firm works with other firms that develop software for a living or consume software that absolutely has to work because it’s vital to their operation. Unlike people who don’t really develop software, we went out to find out what companies are doing. We looked around and said, “If we go out into the jungle and make some observations, what is it we’re going to see?”

When I’m describing the BSIMM in a talk, I use the statement that “monkeys eat bananas.” The idea is that you go out into a jungle and you see a monkey eating a banana. Then you go out into a different jungle, and you say, “Look, monkeys eat bananas in this jungle too.” After making this observation in 30 jungles, you say, “You know what, monkeys seem to eat bananas.” This is an observation; it’s not a value judgment. We don’t say things like, “Don’t steal your neighbors’ bananas” or “Only eat yellow bananas and not green ones” or “Never run with a banana in your mouth.” There are no such value judgments in the BSIMM. Instead, we focus on observation. What is it that we can observe?

The insight was that after we’ve been doing software security for a decade, and after some companies have spent literally millions of dollars and hundreds of thousands of person-hours on software security, surely we’ve figured out something that’s safe. Even though the particular histories or case studies of each of these 30 firms

are distinct and different, we can still use the same measuring stick.

That was the trick: coming up with a measuring stick that applies no matter what the particular history is or what the evolution of the various software security initiatives was.

SOBEL: If a business wants to apply this measurement in its organization even though it hasn’t participated in such efforts before, what does it need to do?

McGRAW: There are two answers to that question. The first answer is quite simple. The BSIMM is available on the website or at the old URL (<http://bsi-mm.com>)—they both point to the same thing now. Businesses can get the model and use it however they want.

Now, of course, when you put out a measuring stick for free and you say, “Use it however you want,” you shouldn’t be surprised if people decide to take your measuring stick, crack it up, build a little fire, and cook a hot dog over it. This is not our intention, but we fully expect that to happen.

You can use the BSIMM to measure yourself, but there’s an issue with self-measurement. Nobody I know who combs his or her hair while looking in the bathroom mirror in the morning says, “Gosh, what a horrible-looking person I am.” Everybody says, “Wow, I’m attractive. I’m ready to go out in the world.” That sort of self-reporting bias is certainly present in the BSIMM as well. To get around that problem, for the 30 firms we have observed so far, in addition to a handful of other people who have assisted with gathering the data, three observers—Sammy Migueles, Brian Chess, and I—have participated in collecting and analyzing the data.

The data is gathered in an objective fashion and then presented to the firm under study. Instead of having a firm fill out some sort of a questionnaire or having somebody with a clipboard ask, “Do you do activity one? Do you do activity two?” we actually have a very in-depth two- or three-hour conversation about software security guided by the Software Security Framework. You can think of this intellectual framework as an archeology grid. We focus on listening. We avoid leading questions, and we try not to elicit responses that align directly with one of the 109 activities. Instead, we’ll say things like, “You have 9,000 developers on your staff; that’s a lot of developers. What do you do to get those people to understand software security?” Then we listen, and, hopefully, they’ll talk about training.

SOBEL: I’ve often wondered whether software developers learn how to apply these measures and assess their software because of training activities targeted toward that business. Or do businesses have to train because their software developers haven’t acquired the skills they need

through whatever program they went through to get their degrees?

McGRAW: Frankly, I don't think that academia does a very good job of teaching students how to develop code in the first place, much less secure code. Computer science curricula focus on theory, how compilers work, operating systems, and maybe some data structures and programming languages. They aren't, by and large, about large-scale software engineering.

I've visited college campuses all over the world, and I give many talks every year to academics. Almost without fail, when I go to a department and I find out who's teaching software engineering, sadly they tend to be the weakest members of the faculty.

I think that's terrible. I wish it weren't the case, and I'd love to pretend that it's not, but it just is. As I have said before, as a discipline, software engineering is lacking. I think that security can help inject some reality, some importance, and some reasonable activity into software engineering. Maybe revitalize the field. To some extent, we've already seen that beginning to happen.

The question is whether or not we should count on academia to teach software security, and the answer is very complicated. There are a number of programs that do in fact pay some attention to that. On page 98 of *Software Security*, there's a list of universities that I think are doing a particularly good job with that.

On the other hand, there are thousands of developers who write code who may not even have a computer science degree. Maybe they learned how to program in a different way. All those people need to be confronted so that they think about software security. Training can help with that, but there's a big difference between training and education.

We look to academia to do education, but we look to our professional workplace to do training. And training requires an education. You have to know how to learn; you have to know how to pay attention and sit in class and absorb concepts. In my view, that's what academia ought to be doing: teaching people how computers work and paying less attention to the skill and craft that is writing code.

Then we have to pick up where education leaves off, providing professional training, mentoring, and getting people to understand security in their everyday job as software professionals. There are many ways to do that, including instructor-led training and computer-based training. Another thing is actually having an interactive development environment such as Eclipse tell you when you're making a mistake.

There are static analysis tools that will tell you when you made a mistake at build time. Maybe a week later, you'll be told, "You shouldn't have done it this way." But we

need to tighten that loop so that we have instantaneous, on-time training. Some of our customers have been asking for that recently, and it's something that we've definitely been pursuing at Cigital.

It's a big problem. This involves a lot of people. If you add up the number of developers in the 30 firms included in our study, it's a very large number of people who are expected to know something about software security (141,715 people, in fact).

SOBEL: I've been interested in the divide that exists between businesses and academics. Do businesses need to play a more active role in getting academic institutions to teach more of what they need, or is it that academics resist change overall? What has your experience been?



Security can help inject some reality, some importance, and some reasonable activity into software engineering.

McGRAW: I rue the day that computer science becomes some kind of a certification program for Microsoft-certified engineers. That would be a complete disaster for the economy and for the world. Instead, it's much better to learn how to think, how to read, how to design, how to elicit requirements, and even some people skills such as how to work in a group (which is tough for some geeks). These kinds of skills are much more important to the future career of a software professional than avoiding, say, certain constructs in C.

Perhaps we should just scrap C and come up with a language that doesn't suck from a security perspective—that doesn't have literally thousands of things that you can screw up. It's a tricky issue. Perhaps some do not agree with my opinion. My undergraduate degree was in philosophy, and all I learned to do was read, write, and think. I didn't learn anything at all about coding when I was getting my philosophy degree. But I've been coding since I was 16.


So the question is, do you actually go to school to learn how to code? I didn't. Many people I know who are professional developers didn't either. What does that mean? These are tricky issues. While I don't have a perfect answer, I certainly have my opinions.

SOBEL: In your experience with the different businesses you've consulted with on security issues, what has surprised you the most? Is it the total lack of security—not even understanding what is basic or necessary?

McGRAW: The most surprising thing I have come to realize is that developers actually want to do the right thing,

for the most part. They actually care professionally about building software that works. Good code. Instead of saying, “Your code is terrible and that’s the ugliest piece of crap I’ve ever seen,” you say, “If you do it this way, it’s better because of security.” Then you get a lot further.

Security people have always had the same trouble that the political left tended to have in the 1980s. That is, pointing fingers and saying, “That’s unfair and that’s broken,” but not actually offering any advice about how to fix the problem. The time has come to tell developers constructively how to build code properly instead of pointing fingers and saying that something was done wrong.



We need to think through robustness, design, architecture, and secure coding while we’re building systems.

Among the many books I have written, about one-half are “bad guy books,” so to speak. Books like *Java Security*, *Exploiting Software*, and *Exploiting Online Games* are about how things break. You might not be surprised to know that those books outsell the “good guy” books about four to one. If you want to make money writing security books, definitely write bad guy books. This is due to what I call the NASCAR effect. People like to watch cars crash, not watch them drive around the track in circles, or go to a car architecture course.

The question is whether we need to teach all developers the attacker’s perspective. Everyone agrees that we need to teach security people and security analysts the attacker’s perspective. We must talk about attacks explicitly, clearly, and loudly. We can’t just classify it and pretend it doesn’t exist. As we all know, what happened in the past 10 years politically leads to incredibly boneheaded approaches to security. We have to talk about how things break, but the question is whether developers need to do that.

Developers love entertainment just like any other human, and they’re susceptible to the NASCAR effect. If you set up a training class that says we’ll show you how to attack code, they’re going to love, it. But the question is whether or not it actually will help them in their job.

Based on conversations with Steve Lipner during the past year or so, I’ve changed my position on that. I used to be adamant about teaching all developers how stuff breaks. Now I’m much more in-tune with defensive programming—how to do it right—teaching the breaking stuff only to the security professionals. That’s a pretty big change in my own philosophy, and that came as a real surprise. Although I’m the guy who wrote *Exploiting Software* in 2004, I have a different view six years later.

SOBEL: Whenever I read something like that, I always wonder whether I’m encouraging individuals to misbehave.

McGRAW: Some people are going to misbehave, and we just have to face the fact that human nature is what it is. I love talking to the press about some of these attacks. The latest flurry of press coverage has been about using social networking to carry out attacks—spear fishing, figuring out your targets, and then maybe even depositing persistent malicious code. These guys say social networking leads to hacking.

If you go back about 100 years or so and look at the early days of the telephone, you’ll see the exact same phenomenon in the press. “Telephone Leads to Murder.” Turns out this person called up his victim and asked him to appear at the murder site before killing him. Now we would just say, “Telephone murder? That’s ridiculous. Everybody has telephones.” Now we’re in the era of social networking and of this computer phenomenon, and we’re still talking about cybercrime as if it’s not just crime.

SOBEL: What do you believe is our biggest vulnerability in software assurance from a business standpoint?

McGRAW: We still have a long way to go in convincing some organizations that the risk-management tradeoffs that they’re making are inappropriate. Some people still tend to focus more attention on features and functions and getting software out the door quickly than thinking about what a bad person might do to make software misbehave.

Part of that has to do with talking about the problem and showing people what can actually happen without too much hyperbole. That’s tough for security people to do. Security people like to pretend to be mysterious and that this is all some sort of very hush-hush rocket science. Frankly, that’s just bull feathers.

These attacks are programs. They misbehave in the way that programs always misbehave. It just turns out that the attacker’s intention is the problem. Rather than trying to put some sort of a magical thing between the bad people and the broken stuff, we need to think through robustness, design, architecture, and secure coding while we’re building systems. The perimeter view of network security is both quaint and ridiculous, and those days are long gone.

SOBEL: Speaking of days gone by, where do you think software assurance for businesses is going to be in the year 2020?

McGRAW: I’m not a very good predictor of the future. I know that if I make some predictions now, somebody might dig this up in 2020 and have a good laugh. So I’m going to equivocate, hedge, and avoid the question instead.

We have made great progress in software security since I got started in the field around 1995. Back then, there were a few good books like Maury Gasser's on system security, and there were some good papers like "Smashing the Stack for Fun and Profit" and Bishop's and Dilger's work on race conditions. Some people were working on software security, but it was a really tiny field. At the time, you couldn't convince anybody that it was important to be working on it. If you look now, I think everybody agrees that we should be doing software security—they're just not sure *how* we should be doing it.

By analogy, we've reached a stage comparable to the way a 35-year-old thinks about retirement. Someone that age might wake up in the middle of the night and think, "Gosh, I wonder if I'm saving enough for retirement." But then he goes back to sleep and forgets about it for another 10 years. With regard to software security, right now people are worried. They know they're supposed to be doing something, but they're just not sure what that something is.

Models like the BSIMM and news stories about how the Chinese hacked Google are helpful in getting people to understand the magnitude of this problem that we face collectively, and the kind of risk that we've painted ourselves into the corner with. But I am optimistic that we're making great progress. I don't see any reason why that progress would just stop dead next year. Models like the BSIMM that allow us to measure are very helpful. To paraphrase Lord Kelvin, "If you can't measure it, how can you do science?"

This notion of actually gathering data and talking about what works instead of just listening to the most vociferous opinion is what we'll see next in software security. I also think that perhaps we'll make some progress in programming languages as well. We have built a lot of incredibly silly things into what we use to build software every day. In some sense, we haven't quite reached the Industrial Revolution. Many people talk about component-based software design, but there's an awful lot of bespoke software development these days.

That's kind of like building a bicycle from scratch with a machine tool and metal parts. There's no standard for how big the wheels should be, so we end up with some bicycles like those old-fashioned ones that have a gigantic front wheel and a little wheel behind (which turned out to be a pretty stupid design). We still have those big-wheel bicycles in software today. Hopefully, in 10 years we'll make some progress toward eradicating some of those problems and coming up with better languages.

SOBEL: Is there anything that you would like to add to our conversation?

McGRAW: The last thing I would like to say is that I'm optimistic, and I think we're making progress. As strange

as it may seem, that's kind of a minority position among security people, who tend to be really cynical, curmudgeonly, grumbly, grumpy people because we're working on this stuff all day. If we retain our optimism and look back at where we've come from over the past 10 years, we can all be proud of some of the changes that we've made. We can look forward to making an equal amount of important change in the future—building systems that are more secure so that when we rely on them, they deserve to be relied on. **C**

Ann E.K. Sobel, a member of Computer's editorial board, is an associate professor of computer science and software engineering at Miami University. She received a PhD in computer science from Ohio State University. Contact her at sobelae@muohio.edu.

Gary McGraw, Chief Technology Officer of Cigital, is the author of Software Security (Addison-Wesley, 2006) and 11 other best-selling software security books. He received a BA in philosophy from the University of Virginia and a dual PhD in computer science and cognitive science from Indiana University. McGraw served on the IEEE Computer Society Board of Governors and produces the monthly Silver Bullet Security Podcast for IEEE Security & Privacy magazine (syndicated by informIT). Contact him at gem@cigital.com.

Call for Articles

IEEE Pervasive Computing

seeks accessible, useful papers on the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing. Topics include hardware technology, software infrastructure, real-world sensing and interaction, human-computer interaction, and systems considerations, including deployment, scalability, security, and privacy.

Author guidelines:
www.computer.org/mc/pervasive/author.htm

Further details:
pervasive@computer.org
www.computer.org/pervasive

IEEE pervasive computing
MOBILE AND UBIGUITOUS SYSTEMS