Silver Bullet Talks with Jim Manico

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.

Jim Manico is the founder of Manicode Security, where he trains software developers on secure coding and security engineering. He’s also the founder of Brakeman Security, which produces a Ruby on Rails security scanner. Jim is a frequent speaker on software security practices and specializes in the notion of building as opposed to breaking. In addition, he’s a volunteer and former board member of the Open Web Application Security Project (OWASP) and authored the book *Iron-Clad Java: Building Secure Web Applications* (McGraw-Hill 2014).

It seems to me that static analysis, which flourished just a few years ago, might be stagnating in the enterprise in some respects. Do you see that in your work? I see some people simply not using it, which to me in 2016 is a shock. I don’t see static analysis as advanced or even interesting—it’s something that everyone should do early in the systems development lifecycle [SDLC]. If anyone I work with isn’t using it, I’ll flag it as a major problem. Even worse, they’ll use it but in very ineffectual ways—like only looking for critical issues or not letting it block anything. I see a multitude of problems: not using it, using it ineffectively, or not understanding the importance of doing it very early in the SDLC.

Do you think lightweight approaches that are now being developed at a number of places will help? We have a lot of problems with moving to DevOps too quickly. People often end up using static analysis to only show critical bugs or remove certain security areas to make it work well within an automated DevOps environment, and then major bugs—that SAST [static analysis security testing] could have found—slip through. I just want it used effectively. The trend I do like is that technology folks are integrating and building static analysis that’s live in the IDE [integrated development environment]. People talked about this 10 years ago, but I see the beginnings of companies who are building it in very effective ways, so it’s like a spell-check for a coder. As I’m writing code, static analysis rules will fire up in real time with advice on how to fix it. That speaks to me.

Tell us about Brakeman, which is a Ruby on Rails static analysis tool. Justin Collins built this open source tool from the ground up and has been maintaining it over time. A lot of people build open source solutions, celebrate it for marketing, and then walk away from it. That infuriates me. Collins has been adding rules to Brakeman consistently for many years now, which is why I’m so impressed with him. I told him, “You should consider a commercial fork of this. I’m not saying stop the open source version, but there’s a need for a pro version.” A year later he did it. Along with Collins and Neil Mattall, I’m a small partial owner and guide of the project. I’m very honored to have a small part in it.

Do you think Brakeman is making a difference in the Ruby community? I do. Brakeman Pro doesn’t have a lot of customers, but those who jumped in are giant. Overnight we were doing business with multi–billion dollar
communications companies that are depending on Brakeman Pro. There’s not a lot out there in the commercial world that does really good Ruby on Rails analysis, and this tool does Ruby on Rails analysis really well. Frankly, I’d like to see more tools that do a couple languages extremely well instead of doing a lot of languages in a less-robust fashion.

**You’re a major OWASP contributor so you must have a lot of thoughts about open source and security and so on. Can you share some of them?**

That’s a big question. I think from an open source point of view, OWASP does a few things exceptionally well and a lot of stuff that we’ll put in the research category. My world is mostly Java, so I like to highlight the Java tools that have been extremely disruptive from OWASP. The main one is the Dependency Check tool by Jeremy Long. It’s destroyed several companies that offer similar services commercially by taking serious income away from them, and that’s what positive disruption should do. It’s innovated very well, it’s free, it works, and it takes out products that normally cost six or seven figures. There’s also a tool by John Melton called AppSensor, which is a way to embed intrusion detection alerts inside a Java application.

Rather than be supported by a large community, I want open source tools to be supported by one or two stellar experts—like the OWASP Java encoder and the OWASP HTML Sanitizer. There’s no good encoding library to stop XSS in Java, so we have Jeff Ichnowski, who wrote the OWASP Java Encoder. There’s also no good HTML sanitizer for third-party-authored HTML snippets, so Mike Samuel from Google wrote the OWASP HTML Sanitizer.

I see these four Java tools as either being production quality or close to production quality, disruptive, or helpful in some way where I could use them in my own projects. I want to point out those tools compared to the hundreds of other “research quality” tools that don’t have that level of production quality. That’s the state of the union at OWASP.

**What about all the piles of open source that security people have to contend with that are outside the security tools range?**

It’s scary because there’s trillions and trillions of insecure lines of code that we depend on right now. Somewhere at this moment a developer is looking for a third-party library to satisfy a programming task on their plate. They find it, they download it, they mess with it for a little while, it works, and now it’s live in their project. But they miss security analysis, even due diligence, in looking at the security history of this library. This developer didn’t look at the code and compound this by millions of times, and that leaves us where we are today. Dependency Check gives me a little something. It’s not perfect, but at least I’m now doing my due diligence as a developer or server administrator.

Let’s talk a little bit about the evolution of Java. Scala has in some sense served as a training ground for new language features. It seems the way they’re trying stuff out in Scala is causing Java to adopt some of those features more quickly these days. Do you think that’s true?

Let me pivot on you. Java 8 and 9 have an extraordinary number of new security features to help Java developers build more secure apps. That’s what excites me about Java. This is the talk that [I gave at JavaOne in September 2016], where I highlighted the security features of the newest versions of Java. They’re trying to improve the cryptography and TLS and get internal validation. So I think they’re learning from the hipster languages, the componentization world, the need for security deeper in the framework. I’m very happy with the movement Java’s making. They’re not as far as .NET, which is another 10 years ahead of the game when it comes to LangSec [language-theoretic security], but Java is getting there and it’s putting in a lot of serious effort.

**What are your thoughts about instructor-led security training versus computer-based training?**

I think a combination of both is going to be important. Instructor-led training [ILT], especially by someone who’s a developer and speaks the language of developers and has good up-to-date material, is a great way to get everyone on the same page. But once the trainer leaves, developers will have questions. So we need to have a center of excellence and
other things available to developers beyond just the training.

My favorite customers go through every slide, make changes, requests, and have their own standards to add to the material. I think those kinds of initiatives where the customer really owns the material and digs into it and asks for changes is how you get the most out of an ILT.

Computer-based training [CBT] scales extremely well. You can't bring in a hundred trainers for a large team, so we often need CBT to supplement what's done in a normal ILT environment. In the past, some of the different CBT stuff I've seen is absolutely horrible.

I'm working with one of your companies, Codiscope, right now to build high-quality CBT training material. It's part of the new trend in microlearning: it's a way for developers to get on-the-spot snippets of training while they're working. Imagine a scenario where an instructor shows up, works with the company to make sure the material is appropriate for what they're trying to accomplish, and then starts using a tool where developers get alerted with these same ideas while they're working and three-to five-minute CBT trainings pop up as they're going. That's going to change how developers write code, and it's the direction I believe the industry should be moving.

Let's go back to OWASP to close things out. Now that the OWASP Top 10 has entered the zeitgeist, what's next for OWASP? What would you like to see?

My recommendation is that everybody is allowed to read the Top 10 once and then they have to be banned from ever looking at it again. Anytime someone is building a software assurance program, they shouldn't be allowed to use the term "OWASP Top 10" at all. Anyone who is delivering training should never be allowed to mention the OWASP Top 10 because there are more than 10 things. I want to replace all global mentions of the OWASP Top 10 with the OWASP Application Security Verification Standard [ASVS]. It's not perfect but it is a 200-item, three-tiered standard on how to achieve basic web-application and (to some degree) mobile- and web-service security. People pay big money to get secure coding standards for their group—the ASVS can help that process. I would never use the ASVS off the shelf as it is. I would fork it and make sure developers own it.

When I'm trying to build a secure coding standard for a group of developers, I don't hand them a standard from security and say, "You have to do this." I sit with them and go over the ASVS one at a time and see which ones the developers want to accept and which ones they don't. Then it becomes a list of standards that they decided was good for their organization. That's a great first step in establishing a standard.