Six Tech Trends Impacting Software Security

Gary McGraw, Synopsys

The past few years have seen radical shifts in the way software is developed, in terms of both process and the technology stack. We must actively track these changes to ensure that software security solutions remain relevant.

Technology, of course, changes—and it changes fast. In the past few years, we've experienced radical shifts in the way software is developed, in terms of both process (CI/CD anyone?) and the technology stack (JavaScript über alles!). As scientists and practitioners, we must actively track these changes so that software security solutions remain relevant.

In this short article, I'll briefly touch on six major technical trends that are directly affecting software security.

Software security has become a serious discipline right under our noses. By my estimate, the commercial software security marketplace, including both tools and professional services, weighs in at around $4 billion or more.

My own firm is typical of industry growth over the years. Cofounded by Jeff Payne and Jeff Voas in 1992 as Reliable Software Technologies and renamed Cigital in 2000, the company originally consisted of only seven software experts. By 2015, Cigital had more than 400 employees. Acquired by Synopsys in November 2016, Cigital is now part of a division with more than 1,000 consultants working full time on software security.

MORE AND FASTER CONTINUOUS INTEGRATION

Continuous integration and continuous development (CI/CD) is all the rage today. Taking agile software development to the next level, CI/CD tightens release cycles to days or sometimes hours. Practitioners sometimes refer to this as DevOps—the idea being that development and operations blur together under so much speed. When software moves that fast, it's hard to take a deliberate, methodical approach to testing, security, and quality. In fact, all of the software "ilities" get dragged along in the CI/CD slipstream whether they like it or not.

CI/CD is especially prevalent among US West Coast

0018-9162/17/$33.00 © 2017 IEEE

EDITOR **JEFFREY VOAS**
NIST; j.voas@ieee.org

firms developing software that lives on or leverages the web. Working on a new app or IoT product? You can bet it'll incorporate cloud technologies, dynamic languages, stateless protocols, and CI/CD. (It'll also create a huge data pile that'll be very attractive to hackers, but more on that later.)

The challenge for software security—code review, penetration testing, and the like—is that good security analysis of the traditional sort takes time. Do you wait until build time to scan your code for known bugs? How long does such scanning take? What if a new build is released every hour? The bottom line is that if security lags, then security loses. We must "automate all the things!" as the meme goes. And that includes security analysis.

CI/CD approaches to quality assurance and testing must include security testing, especially testing based on misuse and abuse cases that can be directly integrated into the constantly running regression test suit. Testing must be fast, observability must be high, and everything must work in the cloud. Are you counting on an API out there to finish your project? How do you test that? Throw in security, and the mountain of challenges gets a lot steeper.

A subset of software security gurus now preach the so-called SecDevOps gospel of speed and automation. In general, this is a good thing and an appropriate reaction to CI/CD and DevOps. But tribal knowledge of software security is weak, and we already see evidence of critical software security practices being left behind: fewer developers are doing threat modeling and architecture risk analysis, and most of those who are lack proper training.

Guess what? If you ignore half the problem (design), your systems are going to be vulnerable in ways that can be incredibly hard to fix later. An automated software security system that spits out dynamic black box testing with a pinch of code review is doable—we've built several—but one that corrects architecture flaws is a pipe dream.

SECURITY ANALYSIS NOW PLAYS WELL WITH THE CLOUD

Despite occasional nose-lengthening claims by its adherents, the cloud has transformed, Pinocchio-like, into a real boy. Developers are migrating to this vast new watering hole in large, dusty herds. This means traditional build-integration and hardcore data-flow analysis (with all software components present) isn't going to be fast enough. We can't simply abandon all the progress we've made in static analysis over the past 10 years, but we also need some lightweight solutions.

New approaches make some aspects of code review faster and easier. The obvious goal is to integrate these approaches into the CI/CD process, which can be accomplished through technical breakthroughs in JavaScript security analysis. Fortunately, despite developers' affinity for the "language du jour," mapping to JavaScript is almost ubiquitous.

This constitutes a sea change in software security analysis, and one that is well underway.

SOFTWARE IS IN EVERYTHING

Predicted for years, the Internet of Things (IoT) is finally here—with a vengeance. Consequently, software is now in everything, from consumer electronic devices to chemical-plant process control systems to the power grid. Does your home's thermostat need software to work? No, but it now contains much software anyway. How about your car? Millions of lines of code. Your TV? It uses software to blab to its manufacturer (and apparently the CIA) about what you're watching (and saying). Getting a new medical implant? It probably has lots of software too.

Ransomware has already hit the IoT, and connected devices are being co-opted into botnets. What's next—people dying because of embedded software? (To those of us in the know, killing people would be nothing new for software; it's just likely to become much more prevalent. Maybe life insurance companies will offer a Skynet rider.)

Before we rush to cram software into everything, we must consider security and privacy. Either build security in, or expect not to have any.

DYNAMIC LANGUAGES ARE CHANGING THE DEVELOPMENT LANDSCAPE

Software containers are a shiny new component in the technology stack that can be helpful for security if used correctly. AngularJS, Node.js, and meteorJS are just a few of the many JavaScript framework flavors out there in dev land.

The new client-side web (which can also be serverless) pushes the computational burden way out to the network edge. But when the edge computes, it also becomes a target for software security exploits. What languages need is security analysis capability that is faster, scalable, dynamic, and cloud based. We can't just give up when our old-school approaches to automated secure code review break down; we must replace them.

Meanwhile, data are piling up everywhere. Everything connected to the Internet produces data, and classifying it all is no easy task. Is it OK for your TV to track what you watch and send that information back to central services? The FTC seems to think so. What about your location data, diligently monitored by your Android phone? Oh, you turned that off? Sure

you did, right up until you needed to use Google Maps to plot a route.

Security system features are already leveraging these data. Modern authentication and authorization systems use lots of factors in often surprising ways, and really modern systems care about context: what you're allowed to do depends not only on who you are but on how carefully your "who I am" binding has been vetted. Is DNA the next factor? Highly likely—bad guys have already replicated fingerprints from social media images.

The real power comes from unifying disparate data into one big database. That's when things get spooky. Privacy barriers will fall away horrifically for consumers and awesomely for vendors. You can believe these conditions won't spawn new attacks if you also believe we can build crypto backdoors that only good guys can use.

AI AND MACHINE LEARNING DEBUT IN SECURITY

Big data aren't really that scary compared to what today's computers can do with some old-fashioned machine learning. I get a kick out of the hype around "deep learning," which is really just a new name for an idea I wrote about 25 years ago. Computers are way faster, storage is way cheaper, and we have way more data to work with, but the technical approach is the same.

When big data meet machine learning, we get some seriously alarming results. Teach a chatbot to use racist words? Unfortunately for Microsoft, that took about 24 hours last year. An AI-driven attack bot? Coming soon to a network near you. Alexa ordering you some cold medicine because it heard you sniffle? Maybe. You actually might need a hacker in the family because the pie drawer in the refrigerator won't open when the camera built into the door and the scale built into the floor trip a red flag with your health insurance provider.

Interestingly, the dusty old AI literature still yields some nuggets. Google "catastrophic forgetting neural networks," put on your attacker-perspective glasses, and sit back and enjoy the ride.

MORE SOFTWARE SECURITY LEADERS ARE NEWBS

The final trend I'll mention here is one of our most important challenges.

At the beginning of this article I mentioned how big the field of software security has become. This is something I'm very proud of, because I've been involved from the very beginning. But with big growth comes a big challenge—those leading the charge in many software security initiatives today are newbs. Oh, they're smart all right, but they might not be leveraging all of the lessons we grizzled old veterans have learned over the years. As just one example, Jerome Saltzer anticipated the architecture/design blind spot in SecDevOps way back in 1973, and, with Michael Schroeder, has since updated his observations (www.cs.virginia.edu/~evans/cs551/saltzer).

We can blame the education system for part of this, but for now we'll have to repeat ourselves. We'll also have to continue to measure results. If you want to know what's going on in software security as applied in the real world today, take a look at the Building Security in Maturity Model (BSIMM; www.bsimm.com).

There are certainly more than six technical trends impacting software security, including blockchain technology, smart contracts, mobile app security at the machine-code level, virtual reality and gaming as an IT security management interface, open source promulgation, the security skills shortage, and device convergence and unification. Software security has come far since the turn of the century, but clearly we still have a way to go.

GARY MCGRAW is vice president of security technology at Synopsys and the author of many bestselling software security books. He also produces the monthly Silver Bullet Security podcast for *IEEE Security & Privacy*. Contact him at gary mcgraw.com.