BLOG @CACM

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3282874

<http://cacm.acm.org/blogs/blog-cacm>

Securing Agent 111, and the Job of Software Architect

John Arquilla describes the new state of cyberspying, while Yegor Bugayenko considers the importance of a software architect to development projects.

John Arquilla
From 007 to 'Agent 111'

<http://bit.ly/2D01wmc>

June 18, 2018

The information revolution has changed just about every aspect of society and security in our time, so it's no surprise that the spy business has been transformed as well. Yes, there are still human "moles" who scurry about inside organizations, gathering up vital information for their foreign masters, and no doubt those "sleepers" deported from the U.S. back to Russia in a 2010 prisoner swap were not the last of their kind; a real-life version of the television series "The Americans" likely continues, in many countries.

Yet adventurous James Bond-like spies have been eclipsed by a new generation of operatives who don't travel the world (not physically, anyway) or drink martinis, shaken *or* stirred. Indeed, most of their time is spent tapping away at keyboards in cool, windowless rooms, their favored beverage some brand of highly caffeinated energy drink. Bond is

giving way to Agent 111 ("007" in binary), who oftentimes might just be a smart bot.

The latest exploit of some Chinese Agent (or agents) 111, made public this month, has to do with sensitive data about American submarine operations. Access apparently was gained by hacking a private contractor doing work in this area for the U.S. Department of Defense. By infiltrating in this indirect manner, cyber-spies were able to vacuum up over 600GB of data that, when the pieces are put together, may provide a valuable picture of how the U.S. Navy intends to operate in contested waters like the East China Sea.

This serious breach, a coup for Chinese intelligence, came in the wake of a string of damaging hacks aimed at strategic targets in the U.S. One of the worst was revealed on March 15 (talk about "Beware the Ides!") in a report issued by the FBI and the Department of Homeland Security that asserted a well-crafted Russian-sponsored intrusion effort had gotten in to our power and water infrastructures. Given these systems are high-

ly reliant upon automated controls, the idea some latter-day virtual James Bond might be able to "cybotage" them is most troubling. For those who worry about how such hacks might hurt our military, give Pete Singer and August Cole's *Ghost Fleet* (<http://bit.ly/2y4v2xC>) a close read.

Back in 2015, one of the things U.S. President Barack Obama and China's President Xi Jinping discussed when they met was the matter of curbing hostile cyber activities aimed at the theft of commercial intellectual property. This Information-Age form of industrial espionage was costing the U.S. hundreds of billions of dollars each year. Both leaders agreed to declare a moratorium on this aspect of cyber-spying, though the Trump Administration has recently charged the Chinese with serial violations to it. Yet it is important to note, of the Obama-Xi agreement, that conducting cyber espionage in the military and security realms was *not* addressed. This omission signaled to intelligence agencies in both countries—and to their counterparts around the world—that a new "cool war" was under way, and it was not to be curtailed.

There are two problems with tacit acceptance of cyberspace-based spying on militaries and other actors. The first is that intrusions, though they may be for intelligence-gathering purposes, are observationally equivalent to attack preparations. How is one to know whether the mapping of one's systems is prelude to an imminent attack, or to an attack at some undetermined time in the future? Either way, this form of cyber espionage is unsettling, because of the threat of actual attack that may undergird it.

The second problem is that the line between military and non-military targets can be blurry, given that much of advanced information technology is inherently “dual use;” that is, the hardware and software that enliven commerce can do the same for conflict. In terms of the Obama-Xi agreement, hackers might legitimately claim in going after sensitive intellectual property—for example, plans to the F-35 fighter plane—that all the tech related to design and production of this aircraft were fair game. Indeed, one need only look at the Chinese knock-off of the F-35 to see the strong similarities, and to infer what happened.

That raises another point about the threat posed by Agent 111: by gaining access to massive amounts of highly sensitive information via cyber-spying, as in this most recent intrusion into the computers of the U.S. Navy contractor, sufficient knowledge may be gained to allow the intruding party to leap immediately to the most advanced technology without having to go through the typically long, repetitive cycles of research, development and design. Thus, Agent 111 is key to a beneficial phenomenon Alexander Gershenkron labeled “late modernization.”

In short, Agent 111 may prove far more effective—and far more lethal, in military effects—than 007 could have hoped to be. Further, cyber-spying is nearly impossible to deter, and when it comes to the views of heads of state, it seems to be accepted, in the context of military and security affairs at least, as “just a new form of espionage.” The only viable answer, given the sorry trail of high-level intrusions into American and other countries’ information systems, is that full emphasis must be placed on improving defenses. Firewalls and antivirals will simply not do. The Cloud, the Fog, and the ubiquitous use of strong encryption should be emphasized as first steps toward mitigating the terrible vulnerabilities that can, thanks to the human and virtual Agents 111 coming on line (literally), hold any nation at grave risk.

**Yegor Bugayenko
The Era of Hackers
Is Over**

<http://bit.ly/20vu1ZX>
July 5, 2018

How efficient is your current software project, and could it potentially

benefit from the addition of a software architect? More importantly, what exactly does a software architect do, and what can they provide to your team? With the world of software development rapidly moving towards more agile workflows amidst democracy in the front seat, the importance of the software architect is understated. A position misunderstood by many is a crucial component that delivers unparalleled guidance in the project pipeline, assigning responsibility to an individual who can turn a company vision into code.

Some might believe the title of software architect is merely a status symbol placed upon a senior coder, signaling a specific level of respect should be delivered; this assumption is wrong. The job of the architect is one that can be highly significant if it is adequately bestowed and the person who receives the title has the qualifications to lead a team. Most importantly, the individual must be able to take the blame for project failures.

The software architect is the individual who takes the blame for when a project fails or is praised when the software, and the team, succeeds. Now, we must understand what is meant when using the word “blame” and why such a large association would be placed with an individual. The software architect is your team’s guide; they are selected to carry the initial vision to a fully solidified working piece of code. As leaders, they elect to take the responsibility for the direction in which they lead their team.

Lead Software Engineer at EPAM Systems Nikolay Ashanin compared the responsibility of a software architect to that of a bridge worker in the 19th century in his published article *The Path to Becoming a Software Architect* (<http://bit.ly/2O3L7ig>) and said at that time the key group of engineers, architects, and workers stood under the bridge while the first vehicles were on it; they staked their lives upon the construction and strength of the structure.

When we say a software architect must absorb the blame for a project, we are merely saying the project outcome that is produced shall fall upon their shoulders. It is entirely up to the software architect to delegate responsibilities of a project utilizing their methodologies, whether that be additional toolsets, their authority, or mentorship and coaching.

Project managers do not always have the option to hire a software architect, as they are typically individuals who are curated by their company, learning and understanding their team over time. In an excellent article (<http://bit.ly/2Ni0wpU>) by Simon Brown of InfoQ, a division of C4 media that focuses on software development, Brown noted, “becoming a software architect isn’t something that happens overnight or with a promotion. It’s a role, not a rank.”

Most importantly, the decision of a software architect must be treated as final. Otherwise, without a true final say in the matter, the individual won’t be looked upon as an authoritative figure. Even a project manager must treat the software architect as the final decision maker when it comes to implementing and producing code. Rather than overruling the decisions of their architect, project managers should seek to replace the individual if product end-visions are not adequately aligning. An individual does not need to be fired, but perhaps placed back within the standard pool of programmers; over time, they might professionally grow to attempt the opportunity once more.

A software architect is the guiding rails for a project; they keep their team of developers moving forward and on-vision while accepting the responsibilities for the team’s actions as a whole. Not only must an architect be able to lead, but also to understand the skills of their team, and how they can contribute to a finished project.

Beyond the ability to craft beautiful code, lead a team to completion, and work under pressure, a software architect must stand as a figure able to accept responsibility for a project; this is the characteristic that defines a true architect. More than simply a senior programmer, more than simply a leader, the software architect stands as a gatekeeper for quality and as a guiding vision for their team. In the end, whether the result is positive or negative, the software architect can stand up and take the praise or blame for what their team has accomplished. □

John Arquilla is professor and chair of defense analysis at the U.S. Naval Postgraduate School; the views expressed are his alone. **Yegor Bugayenko** is founder and CEO of software engineering and management platform Zerocracy.

© 2018 ACM 0001-0782/18/12 \$15.00