Editor: **Robert Blumen**
Salesforce
robert@robertblumen.com

Francois Raynaud on DevSecOps

Kim Carter

THE SOFTWARE ENGINEERING Radio podcast recently added five hosts to the team: Kishore Bhatia, Nate Black, Kim Carter, Matthew Farwell, and Bryan Reinero. They bring new interests, and I'm looking forward to their contributions. The podcast continues to grow in popularity and is projecting more than two million downloads this year.

In episode 288, host Kim Carter sits down with Francois Raynaud, a leader in DevSecOps, which aims to bring practices pioneered by DevOps to application security. Raynaud emphasizes the importance of building security in from the start, because treating security as a "bolt-on" to the end of the process is far costlier and can damage the relationship between security and development teams. Many DevOps principles—such as test automation—can easily be applied to security, and the adoption of these principles can help products and businesses succeed securely.

Portions of the interview not included here for reasons of space include training, mapping the attack surface, the Internet of Things and security, and agile security. Listen to the entire interview at www.se-radio.net, as well as new episodes that have been published since the last column. —*Robert Blumen*

Kim Carter: Francois Raynaud is the founder of the DevSecCon conference. He's actively involved in security automation projects and supporting continuous delivery, and is currently an enterprise security architect for a global retailer. He previously worked at ASOS, Betfair, Verizon Business, HSBC, and RSA. His consulting engagements include implementing computer incident response teams, incident response strategy, security architecture design, IT security management, and penetration testing. Francois, can you give us a quick summary of what DevSecOps is?

Francois Raynaud: DevSecOps is about using the DevOps methodology for security. It's about breaking the silos of security, giving that knowledge to the different teams, and ensuring that security is implemented at the right level and at the right time. DevSecOps puts security at the forefront of requirements to avoid the costly mistakes that come from treating security as an afterthought. Traditional security has always been about exclusion—for example, "need to know" and using the security policy to prevent people from disclosing secrets. DevSecOps is about promoting inclusion and working as a team.

0740-7459/17/$33.00 © 2017 IEEE

For people to embrace an idea and a culture such as DevSecOps, we need to discuss what's defective with the status quo. Can you explain what's wrong with traditional delivery approaches?
With traditional delivery methodology, you get code from the development team and the security team, and then you merge everything into the final build. At the end, a project manager tells the security team, "I completely forgot, but I have this line item somewhere that says 'Get security approval.'" The security team is told, "We have to release this product to make money for the company. Can you please sign off on it?" When you dig into the application and the network after the QA [quality assurance] processes, you realize that no security has been implemented.

Traditionally, security is done after the development team is finished with the product. And at that point you end up with a list of bugs that are difficult to fix. The project manager thinks, "If I implement all these fixes, I'm going to be late and the company's not going to be happy, so let's just forget it and we'll do that in the next iteration."

We want to change the mind-set to include security at a project's inception. We want security to be included in the nonfunctional requirements. We want to ensure that the developer or product manager does not speak only to the development team in initial meetings. We want them to include security. For example, a product manager wants to give customers access to some data without any kind of authentication. Security has always said no to that. But with DevSecOps we want to say, "Yes, you can do this, but you need to do it securely."

Security has a bad name. Let's be honest: we haven't been the most efficient industry. Lots of people made lots of money by creating "bolt-on" tools, but then you end up with Lego-type security, rather than changing things. The aim of DevSecOps is, "Take our knowledge; change it." We want to do things differently. We're here to help; we're not here to say no.

What's wrong with retrofitting or attempting to bolt on security to a project when it's nearing go-live, or even once it's been released, when we get a better picture of where our security defects are?
In a development lifecycle with security at the end, you'll build your applications in a fundamentally insecure way. For example, you've built your logging without taking into account compliance requirements. Or you realize that credit card data has been stored in a text file to make it easy to access. When the product was developed, they put it in an Amazon [AWS S3] bucket accessible by everybody because implementing authentication would have created key-management issues for the development team.

Companies have been completely brought down by this. One example that springs to mind is CodeSpaces. This was a trading company, and they had put not only the application into an AWS [S3] bucket, but also the backups. Unfortunately, they put the encryption key in a public area, and someone malicious got one of the keys and deleted everything. This company closed down in a matter of days. They couldn't recover from this, so they lost everything.

What can DevSecOps practices do to fix this situation?
People will ask security teams, "Why do we need to implement authentication?," and we say, "Because that's written in the security policy." But why is it written in the security policy? What are the consequences of not following it? It's important to explain why we're doing it. We're not just a bunch of people wanting to say no; we've studied security and we need to share this knowledge.

Successful implementation [of DevSecOps] happens when the security team provides knowledge and tools and the DevOps team runs them. There's no reason for a security team to run the tooling as a completely out-of-band management process. Use the tools you have at your disposal already. The CI/CD (continuous integration / continuous delivery) process, for example, is fantastic from a security point of view.

Teaching a security person how to code is much harder than teaching a developer how to code securely. Take me as an example. I can't code properly, and if I wanted to it would take me years and years to arrive there. But when I've trained development teams, they've picked up security really quickly. The [improved] results you get from penetration testing … after security training for developers is really impressive.

How does DevSecOps propose that the relationship between developers and security professionals work?
Start by sitting with each other. I've done lots of incident response and forensics sitting in a glass box, where nobody can actually see what you're doing. Why should you hide everything? Working in silos never works.

Use the methodology of automation for the benefit of security. When [incident response teams] realize attacks are coming against a particular aspect of your website or application, include that as part of the QA process. Give the attack pattern to your developers, so that they can actually change the application accordingly.

Having a "security champion" is one way to do it. This is where the security team teaches one of the developers about security, and then [that person] disseminates the information to the rest of the team. It's really about knowledge sharing.

I used to work in a company that was doing high-frequency trading. They had gamified finding bugs. Two years in, we had five known issues that we wanted the developers to discover. One of the guys came back with 10 of them. At this point we said, "Wow, they got it."

Then, developers get excited. The people who found the issues become your security champions. They get a free T-shirt and can also get certification where we sponsor them to learn more about security.

Have you found that security professionals who aren't integrated into the development team are often regarded by development teams with disdain and lack of respect?

Every day. The security team is normally the team that says no. They don't say anything else. When somebody approaches the desk, they just say "No, no, no, you can't do that—that's really crazy. What kind of idea is that?"

Think about the business. Business is here to make money. The functionalities that the project or product manager is trying to implement—there's a good reason for that. They want to make the businesses more efficient, or they want to get more customers. By making the security team part of the decisions and part of the discussions, you're helping everybody break those silos.

Security should not be separated. That's the key. Forget "need to know," apart from a few areas where we need to restrict information. Give your knowledge to others. It doesn't cost you anything—it's free for everybody.

Can you explain what "shifting security left" means?

That's the big buzzword at the moment. If you start from left to right, as you do in development, and security is bolted on at the end, it's a badge saying, "You've been certified. Well done."

[Shifting security left] is when you start from the nonfunctional requirements. For example, in a financial company you explain [at the start], "We have to think about PCI requirements." That's the essence of it: start from the beginning with all the different teams.

Isn't shifting security left going to slow development and ultimately cost the organization more?

Initially there will be a learning curve, where suddenly this security person is asking lots of questions. But if you think about the costs of implementing security later on, that's completely different. [Think about] fixing a bug in production. That'll cost you a fortune. You'll have to stop production, redo QA, [rebuild] all your artifacts, do all the version control again, and [update] all your documentation.

If you do it at the beginning, once everything is being built, you can reduce these costs. By shifting security left, by discovering issues and bugs at an earlier stage, you can easily incorporate it as part of your QA process. The lag you'll experience will go down, and the cost of fixing security will be much lower.

I haven't seen customers who were not pleased with the implementation—especially the product managers; they just love it. Instead of the traditional penetration testing that occurred at month three, month six, and month nine, it's all built in. You build in your security to make sure it doesn't cost you in the end.

We've had lists of the most commonly exploited defects, such as the SANS Top 25 and OWASP (Open Web Application Security Project) Top 10, since around 2003. The same types of trivial defects are still the most often exploited, but development teams are still introducing those defects to the solutions they're delivering. Is DevSecOps the answer?

DevSecOps is part of the answer. DevSecOps emphasizes threat modeling, which is quite fun. If your development team understands how to do threat modeling, then they can incorporate those tests as part of QA. It's easy. Your OWASP Top 10: incorporate those as part of your QA process. Give developers the ability to test for [those issues] themselves.

When you shift security to the left from the start, you do threat modeling and testing [as part of the development process]. If you don't have any variation by the end of the day, then you know [how many and what defects exist]. If you combine this with metadata from your CI/CD, that'll benefit your [incident response] team. They can dig into this metadata and say, "I have a problem with this server; this is the type of service or software, and this is the level at which it's running." You have the ability to do an incident response really quickly.

Do you have any tips on how you can transform a poorly performing team that has minimal focus on security to a high-performing team with a good focus on security?

Speak to your project and product managers. Your product manager is

SOFTWARE ENGINEERING RADIO

Visit www.se-radio.net to listen to these and other insightful hour-long podcasts.

RECENT EPISODES

- 293—Yakov Fain speaks with host Matthew Farwell about the popular Angular web development framework.
- 294—Host Edaena Salinas interviews Asaf Yigal about applying machine-learning algorithms to the intelligent interpretation of log data.
- 296—Host Matthew Farwell chats with computer scientist Edwin Brady about the Idris language, in which types are first-class citizens.

UPCOMING EPISODES

- 297—New host Kishore Bhatia and guest Kieren James-Lubin provide an introduction to blockchains for developers.
- 298—Moshe Vardi talks about solving computationally hard problems with host Felienne.
- 299—Edson Tirelli and host Robert Blumen discuss rules engines, which execute one or more business rules—such as legal regulations and company policies—in a runtime production environment.

actually quite interested in security. Security has become an added value for companies. They're starting to understand that you can provide great functionality and access to information. But if you provide it securely, that'll be your selling point. Let's be honest here: everything is about selling; everything is about making money.

Can you think of any other benefits of bringing the security focus from the end of the software development lifecycle to the beginning?

As security folks, you are not better [than the developer]. The developer is trying to do his job, and security is trying to do our job. Let's not confront each other—let's work together.

At all the companies where we've implemented DevSecOps, there was some tension at the beginning, which we quickly resolved by making people understand the need for it, how much cheaper it'll be, and how they won't see as much of me, which is always a benefit for everybody.

KIM CARTER is a technologist, engineer, information security professional, entrepreneur, and the founder of BinaryMist. He loves designing and creating robust software and networks, breaking software and networks, and then fixing them and helping organizations increase productivity. Contact him via binarymist.io or follow him on Twitter @binarymist.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscribe to *IEEE Software* by visiting www.computer.org/software.

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.