EPISODE 699

[INTRODUCTION]

[0:00:00.3] JM: DevSecOps is the idea that security practices should be adopted by everyone within an organization. Where the DevOps movement emphasized the breaking down of silos that kept apart engineers and operations teams, DevSecOps emphasizes moving security out of a siloed audit process and distributing security practices throughout the supply chain of software.

In the past, software development usually followed a waterfall development process. Each step in building software was serialized one after another. First, the software was planned then it was built and then it was tested. Finally, the software received a security audit at the end. If a security vulnerability was not discovered during that final audit, it was likely that the software would get released with that vulnerability.

With continuous delivery, we can be continuously checking for security. Every new release can be tested against a battery of automated security tests. The open source libraries we use can be scanned to make sure they're up to date with patched versions. Static analysis can discover memory leaks and buffer overrun vulnerabilities and the systems that you're using to scan your continuous releases for security vulnerabilities, these systems are continuously updating themselves, so you're getting better and better security coverage over time when you are using services that are checking your software for vulnerabilities.

DevSecOps isn't just about the continuous integration release process and how you work security and perhaps testing into that. It's also about bringing security to a broader audience. There's obviously stuff like phishing attacks that everybody in the organization is vulnerable to and need to be made aware of. We talked about those some in this episode.

Edward Thomson is today's guest. He's the Principal Program Manager for Azure DevOps at Microsoft. He was previously on the show to talk about a subtle vulnerability in git, which was a popular episode and I recommend checking that one out. Today, he joins to talk about how an organization can adopt DevSecOps and introduce security practices into continuous delivery

pipelines. We also talk more philosophically about security; defining the most common security risks of a software company today.
We talk about Shadow IT infrastructure, where you have cloud servers and physical servers that people may not have documented, but they're connected to your network. We talk about phishing, we talked about all kinds of security-related issues and it was a really fun wide-ranging episode.

Full disclosure, Edward works at Microsoft which is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:03:14.4] JM: A thank you to our sponsor Datadog, a cloud monitoring platform bringing full visibility to dynamic infrastructure and applications. Create beautiful dashboards, set powerful machine learning based alerts and collaborate with your team to resolve performance issues.

You can start a free trial today and get a free t-shirt from Datadog by going to softwareengineeringdaily.com/datadog. Datadog integrates seamlessly with more than 200 technologies, including Google Cloud Platform, AWS, Docker PagerDuty and Slack. With fast installation and setup plus APIs and open source libraries for custom instrumentation, Datadog makes it easy for teams to monitor every layer of their stack in one place. Don't take our word for it, you can start a free trial today and Datadog will send you a free t-shirt. Visit softwareengineeringdaily.com/datadog to get started.

Thank you to Datadog.

[INTERVIEW]

[0:04:21.7] JM: Edward Thomson, you are the Principal Program Manager for Azure DevOps. Welcome back to Software Engineering Daily.

[0:04:27.2] ET: Thanks for having me.

[0:04:28.5] JM: Last time, we talked about security of git and specifically a git vulnerability. Today is more of a general set of topics and the focus will be DevSecOps, which is a broad subject. The idea of DevSecOps is that everyone is responsible for security. How does this contrast with the past? Who has historically been in charge of security before the idea of DevSecOps?

[0:04:54.8] ET: That's a good question, because in many organizations the answer is nobody and that's really scary. I think what you've seen in many places, people that were taking security seriously was that it wouldn't be baked in at the start. You would get some maybe program managers, or project managers, people to plan the project. You'd get them around in a room and they would plan how work would get done. You'd have some engineers that implemented it. Then all of a sudden you'd at the end of everything have a security review.

Then the security team looks at the terribly insecure mess of code that has inevitably been written, because that's what happens, right? Engineers are pretty good at security. We have a mind to security we hope, but the security team always comes up with really clever ways to break your code. Having that at the end means that you missed the boat, right? You could have baked this in from the start. Instead, you wrote a lot of code and now you're going to go refactor that code, try to fix that code.

There may be some huge architectural assumptions that you made that were bad that led to some insecure designs. Best-case, that was how we approached security back in the day. Even at Microsoft, when I started at Microsoft, security was not something that happened at the very forefront. It happened after planning, but it still didn't happen as early as it could have.

We would go back and forth with the security team to release a piece of software, but it happened late in the game at least. We felt like they were always there for us. We could always ask them questions, but we never did. We had that sign off at the very end game, but it was in many cases too late to prevent a lot of duplication of effort and to prevent a lot of rework. Having it at the end was a bummer.

[0:06:54.2] JM: Well, it's a waterfall. You have a software release cycle where you have this end step of the software release cycle where it gets audited for security and it's this manual bottleneck. As we go from that to everybody in the software supply chain being responsible if we truly follow that notion of a DevSecOps. Is it really everyone who's responsible, or is it just engineers? Is it product managers, the people who are not in the code at all? What can those people contribute to security?

[0:07:28.4] ET: I think it is. If for no other reason, then they are in charge of the planning process generally. Honestly, in many organizations it's the program managers or the project managers who have relationships across teams. If you actually have a security team that is outside of your bubble, of your team itself.

If I look at what we do on the Azure DevOps team, we have security people on our team working on Azure DevOps that are focused on our team, and then we also have a number of security professionals throughout Microsoft. Some of them are coming from Microsoft Research and they're actually working on crypto-algorithms, whether that's maybe elliptic curves, or even quantum cryptography. It's the program managers that know really who to bring in from outside of the group to enable the security discussions. I think that's very, very important. I think it's critically important. Plus they're there to make sure that quality is really baked in from the beginning, right?

Especially an organization that is adopting DevOps, we're adopting a more agile transformation and adopting security into that new way of working, bringing it closer to the beginning. We call it shifting left around Microsoft. We're moving the security into the planning and actual development. They can ensure that that gets done for a team that isn't used to it yet.

[0:09:00.9] JM: It's 2018. What does security encompass in 2018? What are the common vulnerabilities that you see making it into production?

[0:09:10.1] ET: I wish that they were different than they were a bunch of years ago. What we see a lot of is phishing attacks. I wish it weren't the case, but people are still falling for phishing attacks on a regular basis. One of the things that we did on our team was that we started phishing our team. We set up a red team. One of the things that the red team does is send phishing e-mails on a somewhat regular basis.

The first one we did, I remember it, and this was a while ago and you can tell, because the e-mail was inviting people to beta test a new Nokia Lumia – Lumina? I forget what it was called; one of the new Nokia cellphones. As a beta tester, you'd get a free cellphone, and so a lot of people were excited about it, and a lot of people clicked the link to sign up for this beta program, and then entered some details. That was a total red team attack.

We sent around an e-mail and we were like, “Wow. A stunning number of people fell for this attack.” In that e-mail that we sent around, we actually forwarded the original e-mail and a not insignificant number of people scrolled down and clicked the link again a second time and entered their details again. That was when we realized that phishing was going to be a real serious problem, even a technically sophisticated place like ours.

We just continue to do it. We continue to educate our team. We continue to run these red team attacks. I mean, attackers don't have to be sophisticated. Enough people fall for phishing attacks that it seems like, I think it's probably the number one vector that we're seeing.

[0:10:50.8] JM: Now we could talk about security on an org-wide level. I think phishing attacks these are things that you need to train people for an employee orientation, or you need to have six-month or 12-month presentations to the entirety of the organization around best practices for avoiding phishing. We could talk about that, but I think probably it's more pertinent to talk about the software engineering side of security, because obviously, security is such a multi-faceted area.

Since most of the people listening are engineers, it'd be good to delve deeper into security at the application level. If we're talking about changing the security audit process in the engineering context, like making sure your app doesn't get hacked. If we move that process from the security audit at the end of an iteration, what do we put in place instead throughout the engineering process? Do we put security throughout the – do we put security checkpoints at product development? Do we put security checks into continuous delivery pipelines? Where do we insert security, so that it is this omnipresent pressure for people to improve their code on the axis of security?

[0:12:12.9] ET: Yeah. I think that what you want to do is bake it in from beginning to end. For me, that means continuous integration. Pull request validation and continuous integration builds. What we do is we bake a number of security scanning tools in right there. They run right after our builds. When somebody opens a pull request, we'll build it, and then we will run all our unit tests.

In parallel, what we'll do is we'll run a number of security scanning tools. One of those is called cred scan. It's a incredibly simple program. All it does is look for credentials, or things that are likely to be credentials, right? It's basically some reg X's that look for things like begin SSH private key, or begin – or looking for X509 private keys for certificates, looking for things that might be passwords, dollar password equals something.

That is an incredibly important way that we make sure that we don't leak any credentials, because like I said, phishing is important, but attackers can escalate and find new accounts, especially service accounts by finding a data that's checked in, passwords that are checked in, certificates that are checked in. It's incredibly common and cred scan help stop that, right? It's not bulletproof, but it does an excellent job.

Another thing that we do is and you can find a number of these. There are a wasp scanning tools that will look for the common problems, the Owasp top 10, top 25 that are security tools. You can just scan your codebase looking for them. They'll find a number of things that are likely to be to be problems. Yet another even more sophisticated level of scanning is there's a number of tools that will also look at your third-party open source, I guess dependencies. We use one called white source, but again there's several.

What they do is they will actually go through all of the dependencies, whether your NPM, or your nougat packages, or whatever framework you're using. They'll look at the open source products that you use and compare them to a list – a database of known vulnerabilities. Let's say you're using libgit2sharp, right? That's an open source project that I work on. If you're using libgit2sharp, we have security vulnerabilities and we've published that information, and so white source can look at the version of libgit2sharp that you're using and make sure that it's not a vulnerable version.

That means that you don't have to hang out on Github. It means you don't have to hang out on mailing lists, looking to make sure that there are never any security announcements and being very vigilant about it. You should still be vigilant, don't get me wrong, but white source is that next level of protection. The other nice thing it does is of course it looks at not just your dependencies, but your dependencies dependencies.

If you're using libgit2sharp, maybe it's using a version of libgit2 for instance and there's a security vulnerability in that. The libgit2sharp maintainers have been lax. We try to stay on top of things, but you never know. Things slip past. It can actually walk the graph of your dependencies, looking to make sure that there are no problems.

Those are the three tools that we use on every pull request and every continuous integration build, and then we also run these nightly. That gives us a pretty high confidence on every change that we're not introducing something totally crazy, right? It doesn't catch everything, it doesn't fix every problem, but it catches a lot.

[0:16:06.8] JM: Now not every app is really security sensitive. You look at something like Angry Birds. Angry Birds, they need to worry some about security, especially if there's payments involved, or perhaps if they have a password system and any risk of getting your password hacked, the attacker could potentially reuse that password in some other app that you have. Maybe you use the same password to log in to LinkedIn, for example.

Angry Birds, there's always a cost to injecting security processes in your development process. It might slow you down and then you might run out of money, if you're a startup. On the other hand, there are apps that – like Stripe for example, or many things that Microsoft works on where very severe security risk, if there are vulnerabilities, because there's money at stake. It can be dangerous work if you don't have somebody with the security core competency working on it, or if you don't have security processes baked in.

There are people listening at varieties of company types and they're thinking how much do I need to worry about security? How do they make those judgments? How do they define how much resource to allocate to security?

[0:17:25.5] ET: Wow. That's an excellent question. For us, we bake it in early and we go after it aggressively, because not only is it the monetary compensation as you as you noted. A number of products actually deal with money. That's a huge deal. You can pay for as Azure DevOps with a credit card, and so we want to make sure that we safeguard all of that data.

We store user data. We'll store your git repositories on our servers, so critically important for us. As you notice, Angry Birds doesn't have the same security posture. I think it's important to look at the inputs that you're taking from users. What are they? As you know that password is critically important to keep those secure. Any user data very, very important.
Even something as simple as high scores; is it credit card information? No. It could be personally identifiable information. You're probably storing e-mail addresses and names and things like that. Even something as seemingly simple as that could get you in trouble with GDPR risks, if that were to leak out. I think that we often as engineers think that it's not such a big deal, right? The data that I'm collecting is not such a big deal, but I would really challenge you to sit down and at a whiteboard with everybody in your company, at least on a small team rather, everybody on your team, and analyze the data that you're collecting and what it could be used for.

Bring in especially the people that, I don't know, the people that took things apart when they were kids. If you had a clock radio and you took it apart to understand how it worked, you are the type of person that starts to think about systems and decomposing systems and what you can do with them. I think that a lot of people would be surprised by the cleverness that a team, a whole team sitting around a whiteboard can come up with for breaking your system and trying to understand how something that seems simple, like a high score list; it's a name, an e-mail address and the highest score, how you might be able to leverage that into other attacks.

I don't think that you should necessarily go crazy. I don't think you should spend all your money on security. What I do think you should do is really take some time and investigate that and think about that.

[SPONSOR MESSAGE]

[0:20:06.5] JM: I have learned a ton from QCon. QCon San Francisco takes place this year, November 5th through 9th, 2018 and I will be going for my fourth year in a row. I always love going and seeing the talks, and in between the talks I hang out and eat some mixed nuts, chat with other engineering leadership about the latest talks and stuff that they're seeing, the 50 different stream processing systems that we're seeing, the different databases we're seeing.

QCon is a place where senior software developers and team leaders and managers and senior leaders, they all gather together and you have fantastic conversations, you have fantastic presentations. It's extremely high-quality, it’s technical, a lot of it is also cultural and about management.

You can get a $100 off by using the code SED 100. QCon is a great learning experience. The presentations this year include 18 editorial tracks with more than 140 speakers from places like Uber, Google, Dropbox, Slack, Twitter. They are curated high-quality talks and you can get $100 off if you use code SED 100 at checkout for your ticket.

Again, QCon San Francisco takes place November 5th through 9th, 2018. The conference is the 5th through the 7th, November 8th through 9th at the workshops. You can go to qconsf.com to find out more. Thank you to QCon. If you are going to buy a ticket, please use code SED 100 and we really appreciate the sponsorship of QCon.

[INTERVIEW CONTINUED]

[0:22:04.3] JM: We also live in a time where you can effectively outsource a lot of this security, I don’t want to call it paranoia, but security to the cloud providers. Because if you for example build your back-end on top of a managed database, a database as a service and you build your compute layer on functions as a service, or you have – you're using containers as a service and the containers as a service have really well-defined boundaries for where they are interacting. You can have a better understanding of the surface area of vulnerabilities than in the past when you need to manage your own database and you don't have anybody that's looking over you.

The cloud providers, they're incentivized to look over your shoulder and say, “Hey, you're messing up here.” It's like when you push code to Github, if you leave in keys for some service like login keys, there are the cloud providers and Github, I think also run scripts to scrutinize your code and if you have left some key that should be private in a public Github repo, they'll give you a heads up. There is some nice economies of scale where you just get certain security benefits for free, but you certainly don't get everything for free.

[0:23:23.5] ET: That's exactly right. I think that that's also very important, because as you mentioned, let's look at Azure, because that's the one that I'm most familiar with. We're looking at, say the TLS versions that you can use to connect to places and we're ratcheting down insecure ciphers and things like that. Github’s even been more aggressive than that, you mentioned Github. You can only use TLS 1.2, you can only connect with various ciphers over SSH, because they've removed the ability to connect with less secure ones, things that are easily breakable.

If somebody were to sniff your network traffic, you think you're protected because it's all HTTPS, but in fact if you're using a really weak cipher, you may not be as protected as you think you are. The cloud providers really are there to help. Once upon a time, you would have had to have known all of this information yourself, you would have basically had to hire a security personnel, or outsource that information. Now as you noted, the cloud providers are doing a lot for you.

[0:24:29.9] JM: We had a show a while ago with a company called Qadium, Q-A-D-I-U-M, and they were a really interesting company, because they're focused on sprawl, Shadow IT. Basically, if you take a company like an insurance company, they've been around for 20 years, 40 years, 80 years and they have just layers and layers and layers of old infrastructure, old code, it was written by somebody who has died, have been buried and nobody knows why this code works the way it does, but it fulfils its API, and nobody knows where all the servers are. They’re Shadow IT. Shadow IT meaning servers where it's hard to know where they are and it's hard to identify all your network connections.

This company Qadium draws a map of all your IT and it finds all your network requests and stuff and helps you discover these orphaned cloud instances, for example. When I was talking to them, it just sounded like the whole world of enterprises are vulnerable to people entering on some piece of Shadow IT and then using that piece of Shadow IT to tunnel into more sensitive pieces of infrastructure. Have you seen this issue of Shadow IT with any customers?

[0:25:52.0] ET: I haven't heard it called that, but yes. I didn't know that there was an organization that specialized in this. I think that's incredibly clever. One thing that I've seen and I'm not going to name any names, of course, but around – so I live in Europe, I live in England, and one of the things that has been going on here in the UK is something called ring-fencing.

The idea here is that the banks when there was – so in the UK we had the same banking financial problems that happened in the US. One of the things that the UK has done to try to mitigate this in the future is they're splitting out retail banks, so the banks that take care of you and I for our checking accounts and our savings accounts. They're splitting those out from investment banking, the kind that could, I don't know, lose all of the money, right, on a bad trade by investing too heavily in Enron or something.

What you're seeing is these banks all of a sudden having to spring to life, they're splitting themselves out of the bigger organization where all these retail and investment was intermingled. You're seeing these banks have to grapple with Shadow IT as you called it, the idea that all of a sudden they've got to implement their own infrastructure, they need their own active directory servers, or Kerberos, or authentication of some notion and understanding what they used to have is incredibly challenging, because as you mentioned, banks have been around for a long time. That IT has grown and evolved and it's not always a clean evolution. It's sometimes well, we've got an AD server over here, we've got another one over here for redundancy. Sometimes there's some important critical server under somebody's desk and whoops. Yeah, I think that's a very real issue. I think that having an organization, a company that can help mitigate those problems is a huge deal.

[0:27:56.8] JM: When you look at those kinds of massive re-architectures that banks are undergoing, or that insurance companies are undergoing – I went to the DevOps Enterprise Summit a few years ago and it was fascinating to me, because you do have these companies with really, really good businesses that need a serious software refresh, because they need to enable new developers to spin up new infrastructure because they want to be an appealing place for new developers, but at the same time they've got this bevy of issues in the backlog that they need to work on, like for example, splitting up the retail banking side from the investment banking side.

In my conversations that I've had with people, I hate to sound pessimistic, but it is rare to see a success story that has finished basically. I mean, I talked to plenty of people who are in the process and I talked to plenty people who are optimistic about the process, but the story gets told about a DevOps transformation and it seems less like this metamorphosis of a worm turning into a butterfly, and more like just this acknowledgment that you are always going to be in DevOps transformation. You're never going to get out of this thing.

It's going to take so long and that's okay, because you have plenty of partners and vendors to help you out with this process, but have you seen any examples of people migrating fully from their previous state of IT headaches and get liberated into where their insurance company now feels a startup?

[0:29:32.3] ET: Wow, what a great question. I talk to most people who are beginning their journey. What I really need to do is go and follow up. I think that one thing you mentioned is that we're always on this journey. I think that that's absolutely right. I think that as you mentioned, it's absolutely okay, because you're going to continue to evolve, you're going to continue to mature.

To answer your question directly, I can think of one organization and they are financial services company, but they're not a super, I don't know legacy, for lack of a better word, financial services company. They started out in the dot-com boom and then they got big. As you get big, you get enterprising and you get process heavy and waterfall and they very much did.

They've been able to turn many of their systems around. I mean, what I saw them do was decouple a lot of their components. I wouldn't say they went to a true microservices architecture, or anything like that, but they started to decouple components from each other and planning from each other. Teams got more autonomy and were able to deploy independently.

I don't know. I would love to know where they really think they are and how successful they think they are. From the outside, it looks like they have come a long way. They're not doing waterfall, they are doing agile and that's new. They are able to deploy multiple times a day. Certainly, that wasn't a thing that they could do before. I think that they're happy, but I don't think that they think that they're at the end of the road. I think that they think that they're still on the transformation that there's – I know for a fact that they are looking at continuing to look at things to improve and I think that's great.

[0:31:26.5] JM: Yeah. By the way, you see the same thing with a startup that's 5-years-old. If you get to a point where you're 5 years old, especially you've under undergone some hyper growth, like one of these ride-sharing companies, or sharing economy companies that has really hit hyper scale. In that hyper scale, you make sacrifices and five years later, you have to pay down those technical debts that you've built. Oftentimes, the people who borrowed from the future are no longer at the organization and it's everybody deals with these things.

To get back to DevSecOps, because I think what we were touching on there is more a question of DevOps and companies finding their way into a DevOps transformation, or somebody from on high initiating a DevOps transformation and the organization responding to that and that flowing through the organization. Now DevOps describes many different things. DevSecOps is more about the security aspects.

I think a large enterprise often has a CSO, a chief security officer, and their role can be to select products, to buy and identify enterprise-wide problems and figure out how to build versus buy and who's in charge of what and how much can we slow down the organization to insert security issues. If there's somebody out there listening that's a CSO, or they're a VP of security at a large enterprise, how should they think about their role? Let's say the company is in the midst of a DevOps transformation and they also want to have this DevSecOps transformation, what is the role of the CSO?

[0:33:02.0] ET: I think the role of a CSO when you are in this DevOps transformation is really to identify the amount of risk that you are comfortable with, and to make sure that you are finding ways to mitigate that risk quickly, or I guess mitigate security problems quickly, because the faster that you can mitigate a security problem, the less risk you really have.

Let me give you a concrete example, if an attacker gets into your network in a small way, right? They do some phishing attack, they get in, they sneak into a little to a rather unprivileged user account, the question is how quickly you can identify that and how you can make sure that that person has the least amount of privilege is possible, right? That they don't have access to production data, for example, without jumping through some hoops.

By understanding that and by ensuring that you can respond quickly and compartmentalize data, I think that you get a lot more faith in the system, for lack of a better word. As a CSO, you can feel comfortable that as you increase velocity, as you ship faster, deploy faster, that you're still not taking on a huge amount of risk, right? If you're enabling your developers to push changes more rapidly, then you want to make sure that the risk is relatively contained.

[SPONSOR MESSAGE]

[0:34:45.4] JM: Data holds an incredible amount of value, but extracting value from data is difficult, especially for non-technical, non-analyst users. As software builders, you have a unique opportunity to unlock the value of data to users through your product or service. Jaspersoft offers embeddable reports, dashboards and data visualizations that developers love. Give users intuitive access to data in the ideal place for them to take action within your application.

To check out Jaspersoft go to softwareengineeringdaily.com/jaspersoft and find out how easy it is to embed reporting and analytics into your application. Jaspersoft is great for admin dashboards, or for helping your customers make data-driven decisions within your product, because it's not just your company that wants analytics, it's also your customers that want analytics.

Jaspersoft is made by TIBCO, the software company with two decades of experience in analytics and event processing. In a recent episode of Software Engineering Daily, we discussed the past, present and future of TIBCO, as well as the development of Jaspersoft. In the meantime, check out Jaspersoft for yourself at softwareengineeringdaily.com/jaspersoft. Thanks to Jaspersoft for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:36:18.8] JM: I think that one place where DevSecOps can really play a massive role without slowing down the organization is in the continuous integration pipeline, because you can insert these well-defined checks and you can often outsource the checks to vendors, for example. You mentioned white source. That's a no-brainer for some company, for some companies that want to have checks of their open source software. You can just outsource that to white source if you have it in the continuous integration pipeline. How does DevSecOps if you are a security conscious organization, and let's say you've already got continuous integration set up, you've got it 80% of your organization is on continuous integration, how would you change your continuous integration pipeline to acknowledge security?

[0:37:09.8] ET: Yeah. The first thing I would do is just grab every – well not every, but let's not go crazy. I would just grab several free tools and plug them right into my CI pipeline. First of all, I would do fuzzing. I write a lot of native code, I write a lot of C, and so there's huge risks there with buffer overflows, with integer overflows. It's relatively straightforward to set up fuzzing, which just throws random data at your application and trying to get it to crash, because at native code when it seg faults on a buffer overflow, that might be exploitable.

We've seen a lot of really clever attacks from fuzzing. The newest fuzzing tools like AFL will actually instrument your code to understand how the branches of your code are executed as it throws different data at you. It can basically programmatically determine attacks to try to find these buffer overflows and integer overflows that might be exploitable. We build that in to some of the code that I work on and that runs nightly, because it's relatively intensive. That's a no-brainer. A lot of these tools are free.

Another set of tools is just static code analysis tools. That might be coverity, that might be sonar cloud. Those will just look for common programmatic errors that you make. You can run this at during pull request validation, or nightly. A lot of these seemingly stupid bugs might be exploitable as well, right? By just cleaning up your code hygiene and improving your code quality throughout, you are likely to catch a number of bugs that might end up being exploitable, because attackers are clever, right? It seems like a silly bug, but you tie a number of silly bugs together and all of a sudden you've got access to a database. Who even knows?

[0:39:07.3] JM: Well, static code analysis, that has implications beyond just security, because we did a show about this and I think it can find things like memory leaks, where your application could just crash and I mean, that could obviously lead to security issues, but it could also just lead to your entire application crashing.

[0:39:24.1] ET: Sure, sure. Yeah, and an attacker could notice that and turn that into a denial of service attack against you. It's not always just accessing data, exfiltrating data. Denial of service attacks are a real way that people can attack your company.

[0:39:39.8] JM: You mentioned fuzz testing. Is this fuzz testing the same as chaos testing?

[0:39:44.3] ET: I don't think so. Not as I understand chaos testing. I am not the best at chaos testing. My understanding of chaos testing, at least as Netflix described it was we're just going to take servers out. We're just going to disappear a server from a load balancer or whatever and understand how the system fails. Whereas, fuzz testing is more about injecting data that is total garbage and your application knows it's total garbage, but whoops, the way it was parsing your e-mail address in this field when confronted with 7 megabytes of data, it crashes in a really bad way.

[0:40:25.3] JM: Got it. What about more targeted attacks, like SQL injections, or pen testing?

[0:40:32.2] ET: I love pen testing. When we started on –

[0:40:34.7] JM: Explain what pen testing is, for people who don't know.

[0:40:36.9] ET: Yeah. Pen testing is basically trying to find a way into your application. What we do on the Azure DevOps team, we do what we call red teaming. We build a group of security-minded individuals and like I said, it's the people that likes to break things as kids and maybe snuck in. I don't know why. A lot of these people tend to be Australian. Maybe it's something growing up in that convict colony, I don't know.

[0:41:06.6] JM: I noticed that too. That Troy Hunt.

[0:41:08.8] ET: Most the joking, but not.

[0:41:10.4] JM: That Troy Hunt is the leader of the crew.

[0:41:12.2] ET: That's exactly right. It's him that I was thinking of, and he's not on our rep team obviously. Like I said, I'm only mostly joking. What we do is we put together this group of security-minded individuals and their goal is to break into our software. What we actually do on the Azure DevOps team is we send them after the most recent things that we've worked on.

When we created what we call public projects back two years ago when you used what was then called visual studio team services, we were aimed at the enterprise. We were not aimed at open source, or anything like that. Over the last few years, especially within the last few months, we've really wanted to offer our continuous integration and continuous delivery pipelines to open source projects. It's called Azure Pipelines.

That move from everything is private, everything's behind a username and password, you have to – we have tightly controlled security to anybody can now queue builds and view build results. That was a pretty big change for us. When we were working on that, the red team tried to find escalations from public projects into private projects behind the username and password. Or when we offered Mac build pools, so we have hosted build agents in the cloud for Windows, for Linux and for Mac OS.

When we started offering Mac OS, that was a big change for us because most of Azure is just Windows and Linux. This was a big deal. We brought in the normal red team. We have a core of a red team. Then we brought in experts in Mac OS to try to understand how we might be able to break out of containers, or what vulnerabilities we might be opening up by offering these Mac build tools.

The idea behind that is that yeah, they'll just try to break in. The way we actually started that was we didn't just say go and figure this whole red team experience out for ourselves, we actually hired an outside pen testing firm. What they did is they came in and they acted as the red team. They tried to attack our application. The funny thing is that whenever this happens, whether it's our red team or an external red team, they find issues. They always do. There's never been a red team event that we've held that didn't find at least a few security issues.

They found their issues and then they trained us in how to attack our own software. After that first time, we've been doing it ourselves. They were really instrumental in helping us understand how to build a red team, I guess.

[0:43:59.9] JM: Since we're talking about CICD pipelines, have you seen any security vulnerabilities that emerged due to a badly configured CICD pipeline?

[0:44:10.8] ET: Oh, yeah. Again, I'm not going to name names. A lot of CICD pipelines, especially the D part of the CICD pipelines, the deployment have various security tokens, or credentials. To go back, we run white source as part of our builds, some of our projects run sonar cloud as part of our builds. Those actually have security tokens that they need to authenticate with to those various providers. Those need to be kept secret, because otherwise, somebody might be able to view all of your security results, right?

Even in the open source space we've seen this where somebody will open up a pull request that is basically, I don't know, runs ENV, right? The ENV command on UNIX systems will dump all the environment variables. If you have configured your build so that these secret tokens are in environment variables, which makes a lot of sense and you're using a, I don't know, less security conscious, for lack of a better word, a CICD system, then maybe you could view these security tokens.

We've actually seen that happen to open source projects, where people have opened a pull request, got some tokens and then use them for nefarious purposes. This is particularly scary if somehow you could tie into the deployment process. Maybe you could, I don't know, update binaries on an FTP site, or on a distribution site. In theory, an attacker could get their own code running on an FTP site that is owned by you and update a version number, convince people to install a new version that they have actually injected code into. It's very, very scary.

I'm really happy with the way that we do things on the Azure Pipelines team with secure tokens and secure information. It's not just us. I don't want to make it sound like we're the only ones that do this, right? Most I think cloud CICD providers take a very strong stance, I guess on security, but not all of them. We have definitely seen some problems.

[0:46:28.9] JM: Let's shift to talking about culture more broadly. One part of mindset shift for DevSecOps is that you have already been breached. It's the assumption that your infrastructure, perhaps it's a sprawling infrastructure like we mentioned before has already been breached. You can have components of a security strategy that are about preventing security breaches, but you also want to have security strategy that assumes you already have been breached. How does my model for security change if I assume that my infrastructure has already been breached?

[0:47:04.1] ET: What it does is it suggests to you that you need to be very careful about containerization, because like I mentioned, you're going to see maybe a phishing attack where somebody gets into, I don't know, some account, whether that's a developer's account, or an ops person's account. I don't know. I look back. I started out in sysadmin. Way back in the old days, I had – we used UNIX groups to manage a lot of things and I was in the ops group, the admin group, and I could read a lot of files just as my own user, because who would ever get in as me? I'm very careful about security.

That's a very old days way of thinking. Now we absolutely do assume breach. I need to escalate my privileges with pseudo, or another round of authentication in order to see privileged things. You can start adding on extra layers to that as things become increasingly private, I guess; two-factor authentication, things like that. You really do need to assume breach. You need to assume that passwords aren't particularly meaningful.

Like I said, the amount of phishing and other attacks, it's scary easy to get somebody's password, so you need to make sure that you have a multi-factor authentication, that you are containing the more private information. As you are storing more and more privileged information, you need to contain that more and more. For our customer data, it is crazy difficult to actually get access to any server that has customer data on it.

An op team might be able to actually go in and I don't know, turn on a server, turn off a server, but all our data is encrypted at rest. Even logging in, they wouldn't be able to see any customer data. Our engineering teams when they want to deploy, they've got to go through a number of security steps. The old model where we just assumed that I or that you had access to something, I mean, that's very much out the window. You've got to assume that the person using your account might not be you, might be a bad actor.

[0:49:20.9] JM: What kinds of tools are you working on at Microsoft that are based around DevSecOps?

[0:49:27.4] ET: The big one that we just released actually – we didn't just release it, we just put it into a private preview. It's now made available to the public, but in a limited capacity. It'll be generally available to the public pretty soon. We have a set of security tools that plug into the CICD pipeline. The big one there is cred scan, so we're making that available to the general public. I mentioned that it was simple and it is, but in that simplicity is a lot of power and it catches a lot of things.

Like you mentioned, Github will notify you if you check in something with something that looks like a password. Now we will as well. You can opt into this as part of your CICD pipeline, or as a part of your pull request experience. I think that that is again, it seems so simple, but it catches so many problems right before they even enter the codebase.

[0:50:24.3] JM: It seems like you're catering to a heterogeneous customer base. You have on-prem people, you have hybrid cloud people, you have people that are entirely on the cloud. When you're building security products, how does that affect what you're giving them, the fact you have this multiplicity of different customer types?

[0:50:44.4] ET: Oh, yeah. We absolutely have all of those, right? It's surprisingly similar. I mean, when you look at it, there's probably some differences between SQL Azure and on-premises SQL server. There's probably some differences between Kubernetes in your environment versus Azure Kubernetes Services. Fundamentally, there are fewer differences than you would think that the fundamentals are what's really important; keeping credentials safe, limiting access, containing the amount of access that somebody can get.

The wrinkles in technology are they're not so bad. There are abstractions on top of these things. Once you're dealing with I don't know, node and dotnet core and all these different application types, you very much get in the habit of building an orchestration system, building a pluggable scanning system, and then having these little bits that plug into them to handle each individual case. It sounds like a lot of work, but to be honest with you, the on-prem versus hybrid, versus cloud is not fundamentally that much of a problem.

[0:51:58.1] JM: To wrap-up the interview, I want to ask you a question synthesizing the last two interviews we've done. We talked last time about git and an open source vulnerability. DevSecOps is certainly an issue that impacts every enterprise that's out there. Does DevSecOps also impact the management of open source projects?

[0:52:20.1] ET: I think that it does. I get to look at it from both sides. I get to look at it from the Azure DevOps side, where we're building a cloud service, and I get to look at it from the open source side where I work on a number of open source components that are surprisingly old-school. We ship code that you install on your computer. That is not something that most people are building software for anymore.

We take a lot of the learnings from DevSecOps, for lack of a better word, and we do apply them to the open source world. I work on libgit2. I don't really work on git, the actual command-line application. I work on libgit2, which is a slightly different branch off of that family tree, but we absolutely are baking security components into the CICD pipeline. Like I said, we're bringing fuzzing tools and we're bringing static code analysis and we want to catch security issues as early as possible.

We've actually had a number of security releases recently, just point releases that are only addressing security issues, because we have been maturing our fuzzing so much and we just keep catching these little problems that had been lying dormant. Now, we're basically burning down that security dead, but we're also catching new problems before they enter the code base, so I guess we're in our own DevOps transformation, if you will there.

[0:53:46.9] JM: Edward Thomson, thank you for coming on Software Engineering Daily. It's been great talking to you.

[0:53:50.1] ET: Thank you.

[END OF INTERVIEW]

[0:53:54.3] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]