EPISODE 826

[INTRODUCTION]

[00:00:00] JM: The modern software supply chain contains many different points of distribution. JavaScript frameworks, NPM modules, Docker containers, open source repositories, cloud providers, on-perm firmware, IoT, networking proxies and so much more. With so much attack surface, securing a large enterprise is an uphill battle.

Jeff Williams is the CTO at Contrast Security, a company that makes infrastructure monitoring tools. Contrast Security works by intercepting networking traffic at a low level and assessing whether that traffic maps to a common threat model. Jeff joins the show to talk about different approaches to monitoring and securing large infrastructure deployments.

The FindCollabs podcast is out. This is a new podcast that I started to talk about the community within the new company that I'm building, which is called FindCollabs. You can go to findcollabs.com to see that product. We're also hiring a React developer. The details for all of these FindCollabs facts are in the show notes for this episode.

We are booking sponsorships for Q3 for Software Engineering Daily. If you are looking to reach 30,000+ developers, you can find more details at softwareengineeringdaily.com/sponsor, and there are some other links in the show notes to updates in our ecosystem.

With that, let's get on with today's episode.

[SPONSOR MESSAGE]

[00:01:43] JM: Testing a mobile app is not easy. I know this from experience working on the SEDaily mobile application. We have an iOS client and an Android client and we get bug reports all the time from users that are on operating systems that we did not test. People have old iPhones. There are a thousand different versions of Android. With such a fragmented ecosystem, it's easy for a bug to occur in a system that you didn't test.

Bitbar is a platform for mobile app testing. If you've struggled to get to continuous delivery in your mobile application, check out bitbar.com/sedaily and get a free month of mobile app testing. Bitbar tests your app on real devices, no emulators, no virtual environments. Bitbar has real Android and iOS devices, and the Bitbar testing tools integrate with Jenkins, TravisCI and other continuous integration tools.

Check out bitbar.com/sedaily and get a free month of unlimited mobile app testing. Bitbar also has an automated test bot, which is great for exploratory testing without writing a single line of code. You have a mobile app that your customers depend on, and you need to test the target devices before your application updates roll out.

Go to bitbar.com/sedaily and find out more about Bitbar. You get a free month of mobile application testing on real devices, and that's pretty useful. So you can get that deal by going to bitbar.com/sedaily. Get real testing on real devices. Get help from the automated test bots so that you have some exploratory testing without writing any code, and thanks to Bitbar. You can check out bitbar.com/sedaily to get that free month and to support Software Engineering Daily.

[INTERVIEW]

[00:03:53] JM: Jeff Williams, you are the CTO at Contrast Security. Welcome to Software Engineering Daily.

[00:03:58] JW: Hi, Jeff.

[00:03:59] JM: I'd like to start on the topic of the software supply chain. Describe the modern software supply chain as you see it.

[00:04:07] JW: Well, that's a great question. It's really complicated. The simple way of thinking about it is folks are building some of their own custom code and then they're also getting some open source libraries. Those are the ingredients that go into an application. But if you double click into those things, they get really complicated.

On the open source side, you've got to look at each open source library has a bunch of dependencies and they have dependencies, and that goes on down for many levels. Then there are some complexities there, because they might be old versions of those dependencies. There's some complexity there because then you got to start looking at, "Well, what's the pedigree of the code that's coming through those channels? Who built it? How did they build it? How do they test it? What tools did they use to test it?" You can even think about are those components possibly Trojan in some way?

I wrote a paper on this at Black Hat a few years ago. We get into it if you want, but it's really pretty easy to backdoor software. So then you got to start looking at the tools that they use to develop that code, like what environment did they build it in? Some laptop with some random downloaded tools, and are they also playing Flash games at night on that same laptop? I don't know. There's a million aspects to the integrity of that supply chain and none of it is very good.

There's also the same thing on the custom code side, because – I don't know, people build code on their laptops even for big companies. They build it on laptops that they use for other things that could easily be compromised. So how does that affect the trustworthiness of the code that comes out of it? You can even look at build tools like Jenkins and so on and imagine like, "Well, what is the code was clean but at some point during the compile process somebody was able to slip in some Trojan code?"

One of the things that I did was I thought, "Hey, what if somebody committed a malicious test case?" Nobody looks at the test cases, but those test cases run on the machine that's building the code. So could it Trojan the actual binary? Sure.

[00:06:12] JM: As you're looking at this distressing array of issues in the supply chain, you founded a company. So you can focus on any of these different areas or you can focus on all of them. I think you're probably more in the direction of the later. But if you were to pick some particularly bad, particularly acute pain points, particular vulnerabilities in that software supply chain that concern you the most, what would those vulnerabilities be?

[00:06:44] JW: Yeah. So probably the biggest thing is the most obvious is don't use components that we already know have vulnerabilities in them. So there's a bunch of them.

Researchers out there do great research and they find vulnerabilities and publish on a CVE, and we know about those. So the simplest thing you can do is just not use ones that have no invulnerabilities in them. The problem is we're probably really only scratching the surface, because all these research is done by volunteers. There's no organized teams that are out swarming on open source libraries. So that's a real challenge for folks.

But I do want to zoom out a little bit and say that those open source vulnerabilities are important, but they're just one piece of the puzzle. Most vulnerabilities are in the custom code of applications. So it's something like 70% or 80% of the vulnerabilities in a given application are going to be in the custom code, and that's a really important piece of your overall vulnerability picture to consider.

[00:07:54] JM: Now, if you're talking about that custom code, that code is closed source. How do attackers find vulnerabilities in that custom code?

[00:08:05] JW: Yeah. So they're really creative about this, and that's why see so many spectacular hacks that happen. I spend many years as a penetration tester doing this kind of work, and really, you take what you know about the application and you start exploring it and testing it and poking it and prodding it and pretty soon something weird will happen. Then you poke and prod it some more and you isolate what's going wrong and you can sort of reverse engineer these vulnerabilities by attacking them. For open source it's a lot easier. I'm a big fan of manual code review. So for things where you've got the code, that's great. You can just analyze it and look at it that way.

[00:08:46] JM: When I think about your background as a penetration tester, it makes me think about what Contrast Security does, which to my mind is you kind of put an agent on the different areas of a software stack and it seems like that agent is doing pen testing at those different layers. Is that a fair way to describe your company?

[00:09:10] JW: It's a little bit like that. We use the knowledge from all the pen testing in the product, but basically what we're doing is passively observing the application as it runs. So where as the pen tester is poking and prodding at it, what we do is much more like what a new relic or app dynamics does for performance. We sit back and we watch. We observe data as it

flows through the application, and let's take a SQL injection example. If we see some untrusted data flow in and make its way through the application and get into a SQL query without being escaped or parameterized in some way that would have stop SQL injection, we know that we've found an exploitable path through the application, and then we report that as a vulnerability. So it's a little bit like pen testing, but it's fully automated and sort of passive.

[00:10:00] JM: Let's go deeper on the SQL injection example. Describe how a SQL injection vulnerability is commonly exploited.

[00:10:10] JW: Yeah. So first thing you have to understand is how SQL injection works, and it's relatively simple. If you take untrusted data from an HTTP parameter, a cookie or a header or something and put it into a SQL query by concatenating it into the query itself, you give the attacker the opportunity to change the meaning of that query. So for instance, if your data is landing inside a coded string, the attacker could send a close quote and then add on some commands. Sometimes you can chain commands, which would allow them to sort of take over the database. Sometimes you have to work within the syntax of the query itself to steal data and so on. So to exploit them, attackers have to figure out sort of where are they landing in the SQL query and then what's the right syntax in order to take over the query and make it do your own bidding?

[00:11:01] JM: The lifecycle of that SQL query malicious injection spans from the user's browser to the application layer on the backend. If it's some kind of microservice, maybe tend it off to another microservice, then it's eventually handed off to perhaps some JDBC driver, and then the JDBC driver hands it off to the actual database where the SQL will actually inject. In that trace of different areas that you could detect on perhaps intercept that SQL injection, where is the best insertion point for your monitoring tool?

[00:11:47] JW: Yeah. So you've latched on to the key problem, is it's you can really detect SQL injection that well if you're only looking at a piece of that trace. So the most complicated part of the data flow is in the actual custom code itself. So at the data flows in from untrusted source and then winds its way through the code and makes it to a SQL query that get sent to the database. So that's really where most of the – Most flows don't end up in SQL queries. So you can't just say every untrusted input in a SQL injection. You got to find the ones that actually

make it into a SQL query. So I think the best place to do that is inside the web application or web API itself.

Now you mentioned the case of microservices where they're all threaded together, and generally my advice is to make sure that all your microservices are independently protected against SQL injection, because you never know how they're going to be structured or deployed in the future. You never know which ones are going to be publicly accessible and so on. So you don't want to count on service A defending against SQL injection for service B.

[00:12:59] JM: Now you want to be able to scan the traffic at different areas of the stack. What does that actually entail? In an implementation sense, if I want to integrate with your security solution and I wanted to scan all my traffic and find vulnerabilities, but I don't want to have to spend a bunch of time integrating with this solution. I just want some sort of like seamless integration thing and I don't want it to lower my performance. What's the, I guess, the different areas of the stack? How do you accommodate that problem?

[00:13:39] JW: Yeah. So I'm trying to rid the world of the concept of scanning. It's almost always not the right way to find security vulnerabilities, because scanning – By definition, it means you're kind of working from the outside and trying to reverse engineer what's going on inside, and it's really difficult. That's why pen testing is so hard.

So to me, the right thing to do is to get inside the running application and monitor it continuously from within. So the way that contrast deploys – I'll give you a Java example, but it works the same for .NET, Node and Ruby and Python and .NET Core and things like that. You take our agent, which is just a JAR file and you add it to the way that you launch the Java command. So you add the Java agent flag, and then you just run your application as normal. As the application starts up, Contrast will instrument it with what I'll call microsensors. They're really just tiny bits of code that take a snapshot of security relevant behavior as the application runs. So then as your user application, as you do your normal development, you do your normal QA, your normal deployment, Contrast is there in the background turning all of your normal QA activity into security testing.

[00:15:00] JM: The adaption of things like containers and cloud infrastructure is uneven across the industry, and the deployment models and the maturity of different enterprises in their migration perhaps from a monolithic architecture in one language to a microservices architecture in a variety of languages. That is uneven.

So I think about the problem you're solving, you're trying to solve. You have solved to some extent as this really big end-by-end matrix of different areas where you need to assess application security. How do you approach that problem without just like getting overwhelmed by the scope of the different areas of the kind of software runtime fabric that you want to integrate with?

[00:16:06] JW: Yeah. Well one thing that we do is we focus pretty exclusively on the application layer. So we don't actually care where you deploy your application. You could deploy on containers or cloud or elastic environments or even in an internal data center. For us, the kind of security that goes on in the infrastructure layer is very different than the kind of security that goes on at the application layer. The infrastructure is all built with the same kind of stuff, and predominantly, to secure it, you're hardening and patching and firewalling and stuff like that.

But at the application layer, security is really different. Every application is this beautiful and unique snowflake that's got all these custom code and it works differently from everything else. So we focus on that problem, and you still do need to secure the rest of the fabric, but what we're focused on is really just the workloads, just that business logic piece. That simplifies it dramatically.

[00:17:05] JM: Let's go deeper into that application layer. What exactly are you doing at that application layer that you're focused on?

[00:17:14] JW: Traditionally, folks use tools like static analysis and dynamic analysis to find vulnerabilities in application. They use web application firewalls to try to prevent them from being attacked. Those technologies came out in the early 2000s. They really haven't evolved to keep up with modern software, and they don't work very well. They're super inaccurate.

So what we do is we're taking advantage of all the contextual information that we can get from inside the running application, things like the code itself, and the HTTP traffic, and the dataflow, and control flow, and all the libraries, and how they're used, and all the configuration, and the backend connections. We can see all of that kind of information and use it to very accurately identify vulnerabilities, and we do it in real time so we can give this sort of super accurate feedback to developers as they code so that they can fix things as part of their normal work, the way they normally fix other kinds of issues that are in their software.

[00:18:19] JM: Tell me more about the actual implementation of it.

[00:18:22] JW: Yeah. So we figured out ways to instrument all those languages and platforms that I talked about, and then on top of that, what we do is we've essentially translated the different kinds of vulnerabilities that exist in AppStack, things like SQL injection, but also things like command injection, and CSRF, and XSS, and XXE, and expression language injection, and [inaudible 00:18:48] objects and so on. We translated each of those into a series of calls. We call it a rule, but essentially it's like what is the behavior of the application that causes that vulnerability to exist.

So for SQL, it means untrusted data flowed into a query without being parameterized or escaped. For something like XXE, it means you parsed an XML document and processed the doc type header. You didn't disable doc type processing, and so on.

Then what we do is we put sensors into the application that allow us to see whether that behavior is happening. So we get a series of events from our sensors all within the agent, our agent, the Contrast agent. When we see a pattern that represents a vulnerability, we immediately report to our team server and alert the folks that need to know about it through the tools they're already using.

[SPONSOR MESSAGE]

[00:19:57] JM: We like to explore cutting edge and the real world solutions enabled by those technologies on Software Engineering Daily. Amazon re:MARS is a chance to hear directly from the leaders and innovators who are using artificial intelligence to shape the future. Amazon

re:MARS is a new global AI event on machine learning, automation, robotics and space. Re:MARS is inspired by the exclusive MARS Conference. It will combine the latest forward-looking science with practical applications. You'll hear from thought leaders across science, academia and business.

Speakers include Amazon founder and CEO, Jeff Bezos. Landing AI founder and CEO, Andrew Ng, and more. You'll learn best practices that you can implement immediately through more than 100 sessions. See demos of automation technologies, engage in hands-on labs, and interact with real robots at the robotics showcase.
When you leave, you will be up to speed with the latest AI trends and best practices. You'll also have the technical skills to apply AI to your own work. I am only comfortable using that term AI because I do think this is a great place to apply machine learning to hire level applications and build applications in these different domains. So I think AI is a fitting abstraction to describe this conference as being focused on.

Amazon re:MARS will take place June 4th through 7th in Las Vegas. You can register today and get $400 off the ticket price. Go to remars, R-E-M-A-R-S.amazon.com and register using the promo code SEDAILY. I am really tempted to go to this conference. It sounds really cool. I am not sure if I'm going to be able to swing it, because I've got a bunch of conferences to go to this summer already. But I will definitely be thinking about it a lot.

So that URL and promo code again is R-E-M-A-R-S.amazon.com and you register using the promo code SEDAILY. Thanks to Amazon re:MARS for being a sponsor of Software Engineering Daily, and this conference sounds pretty sweet.

[INTERVIEW CONTINUED]

[00:22:30] JM: I see a couple areas of engineering we could dive into here. There's the agent that you've implemented, you've built, and there's the sensor. Can you go a little bit deeper into each of those?

[00:22:43] JW: Yeah. So the agent is just our way of instrumenting the application and it also has a rule engine that processes the events to see if there's been a violation of our rules. The

sensors themselves are really just callbacks. They're code that we insert into the methods of the application into a security relevant methods of the application that just give us some visibility into what's going on there. Basically, we snapshot the parameters to the method. Some of the state and the return as well as the stack trace at each event.

Then, if we notice a violation – We don't really report the violation until the end of the trace, like the data flows through, it bounces around, we get a whole bunch of events. Eventually it makes it into that SQL query, and then we go back and we verify the trace. We say, "Hey, did this trace have the proper validation or escaping?" If it did, then we just throw it away. There was no violation there. We don't have anything to report. But if there was a violation, that's when we turn it into report that we send up to the team server as a vulnerability.

[00:23:53] JM: So this sensor – Explain what a software sensor is in your context.

[00:23:59] JW: Yeah. Imagine if you're instrumenting an application with log messages. That's like a really simple way of instrumenting the app, and what you're doing is you're actually weaving those logging calls into methods that you care about. Contrast is a little bit like that, except for instead of a log message, we're actually weaving in the code that just takes a snapshot of what's going on in that method and reporting it to us as supposed to putting it in a log file.

[00:24:25] JM: Does the sensor read like JVM bytecode or just like whatever runtime environment is – If you're running Ruby code, you've got like the C struct sitting in-memory. How are you reading this runtime?

[00:24:47] JW: So we're actually weaving these sensors into the code of the application. So as the binary code of the application moves form disk into memory, we hook that and add in our callbacks, our sensors, into that code so that it ends up looking like the developer put that code in originally, but we're just weaving it in at runtime.

[00:25:11] JM: I assume that's pretty lightweight weaving in of code. It's not going to really impact performance in a measurable way.

[00:25:20] JW: In development, there's a little bit of a performance that we're doing quite a bit of work. Full dataflow analysis of complex. We put a lot of work in the performance. So typically we add about five milliseconds to a round trip request in development. But one thing we haven't really talked about is Contrast also protects applications in production. So using the same technique, we can also identify real attacks on applications.

When we see an attack, we prevent it from exploiting the application. That's called RASP, or runtime application self-protection. That is much faster. There we add about 50 microseconds to a round trip request. So about a 20th of a millisecond. That's about 20 to 40 times faster than a typical WAF, because we're doing it right within the code and not taking an extra hop and not doing a whole bunch of extra parsing and stuff.

[00:26:17] JM: Just so we convey the architecture again to people who missed it here, let's go through an example. So give me an example of a vulnerability, it gets detected by your code weaving sensor and the agent gets notified and the agent manages to propagate the correct commands in order to stop that. That's so complicated. So tell me – Like just give me the end-to-end there.

[00:26:43] JW: Yeah. I don't know. Maybe I'm used to it, but, really, it's not that complicated once you get used to it a little bit. Any more than like doing performance monitoring is complicated. It's really essentially the same thing. We're just doing it for security instead of performance. But let's zoom out real quick for just a second.

Imagine you're a software development project and you've got – I don't know, a dozen developers working in various environments, dev, and you got a pipeline with some CD servers, some test servers and you push into production. The way you would deploy Contrast is you'd take our agent and add it to your development machines, because they're going to be running the application locally. You'd add maybe your CICD server and your QA server because you want security testing there as well. Then you can also put it in your production servers set to be in protect mode.

After that, that doesn't take for very long. It takes like 30 seconds to add the agent to an application. From that point forward, all you do is your normal development process, you just

write code, you test it locally, you push to your GitHub or whatever, you do automated builds. All that just works normally. Contrast is always in the background using this instrumentation-based technique to just identify if your code steps out of line. If it has any behaviors that represent a vulnerability, Contrast will identify it.

Then what Contrast does is it sends all those vulnerabilities to our central team server that's either SaaS or on-premise and that team server then is able to notify the people that need to know about that vulnerability through whatever mechanisms you set up. We got a ton of integrations into tools like IDEs, like Eclipse and Idea and Visual Studio. We integrate into Slack, Jira, GitHub issues, a bunch of Azure tools, tools like Jenkins and other CICD servers and even into SIEM systems, like Splunk and ArcSight and so on.

So we're really trying to give you great visibility into application security across the whole lifecycle. But what it really shines is if you're a big enterprise with hundreds or thousands of applications, and now instead of having to go scan each one of them individually and get an expert team involved slowing down your pipeline and generating tons of false positives and causing everyone to hate security, instead, now, all your applications are sort of testing themselves in parallel. All the results are coming together in your dashboard so you can see exactly what security you have across your whole enterprise all at once.

[00:29:29] JM: Can you describe the communication protocol between the sensor and the agent in a little more detail?

[00:29:39] JW: Yeah. So the sensor to the agent is all within the application space itself. So the sensor simply makes a callback into the agent code. So that's all local within the running application, and you need that, because you want that to be super high-performance.

Now, the agent connects back to the team server to report – It just reports vulnerabilities and things about libraries and things about attacks and things like that, but that is just a REST API and it's mutually authenticated SSO REST API outbound connection.

[00:30:15] JM: So if you detect something, how do you prevent it? Maybe I miss this. Where are you halting a command from executing?

[00:30:26] JW: Right. So a web application firewall would just look at the HTTP request and say, “Oh, I think that’s an attack, so I’m going to stop it.” The problem with that though is that you never really know if it’s an attack or not just by looking at the HTTP request. Many JSON requests and XML documents kind of look like attacks. So WAFs over-block and under-block can make tons of mistakes.

Contrast works differently, right? So we can actually see that data come into the application. We can watch where it goes. Imagine an attack, like someone put in single tick or 1=1 into a web form, and that data flows through the application and makes it into a SQL query, changing the meaning of that SQL query. Then Contrast can intervene and say, “Hey, we’re going to not send that query to the database. We’re going to throw an exception and jump around that code so that the attack is diffused.”

[00:31:23] JM: So just throw just whatever, IO exception, or virus injection exception or something.

[00:31:31] JW: Well, we want it to look like a normal SQL injection, a normal SQL exception, because we want the application to be able to recover, right? If there was transaction in process or something, you might want to roll that back. So we have this – Our Hippocratic Oath is never break an app.

[00:31:47] JM: Right. Okay. So you can just throw a general exception and through the inheritance hierarchy, you’ll get the right exception.

[00:31:54] JW: That’s right. That’s right.

[00:31:56] JM: Very interesting. So one thing I have seen in conversations with different companies about enterprises is that enterprises have this IT sprawl, where oftentimes they have so much infrastructure that they cannot even like find all of it, or they don’t know where all of it is. So if you’re integrating with a company, how do you find all of their infrastructure in order to get instrumented?

[00:32:26] JW: Yeah, that's a tricky problem, and there are some companies out there that focus on that. We'll call it the inventory problem. I would say you can't really secure anything that you can't get your hands around, right? Like the first step is to know what you have to secure.

We try to make that easier a couple of ways. One is we try to make it really easy to make Contrast part of your default install. So we've got fantastic integrations with a bunch of different cloud environments. Like Pivotal for instance, we got a great tile and a lot of enterprises use our Contrast Pivotal tile to just make that part of their standard deployment. So whenever they deploy something, Contrast goes with it.

Then when the application starts up, Contrast will connect back to its team server and start sending vulnerability data and running attacks and doing all its work. So I think that's one way to try to get a handle on all your applications, but you could be running – There could be some web app running under some desk somewhere that we're not going to know about. Solutions there, you can do things like networking monitoring to see if there's web traffic going around. You can do other kinds of scans to try to flush that stuff out. But we're kind of like step two. Once you find it, then you want to bring it in and make it part of the ecosystem so that you're tracking it and protecting it forever.

[00:33:44] JM: The amount of vulnerabilities in the world is quite numerous. I think about the Apache Struts vulnerability that managed to leak everyone's Equifax data.

[00:33:58] JW: Yeah.

[00:33:59] JM: In an ideal world, you would be able to have insight into all the nuances of all the software that is running across an organization and an up-to-date understanding of the vulnerabilities in those respective pieces of software, from SSL, to Apache Struts basically. That seems kind of hard to do. What's your strategy for maintaining an up-to-date, I guess, compendium of vulnerabilities and then rolling out that correct instrumentation to your agent and your sensors?

[00:34:38] JW: Yeah. So you're right. You can't chase CVEs, because there's half a dozen every week of new that somebody needs to get worried about. So Contrast's strategy is to fundamentally prevent whole classes of vulnerabilities from being exploited. So that Equifax breach was an expression language injection problem. Basically, they're double evaluating an expression using user input, and so the attacker could use that to send in an expression that contains native code, runtime.exec command.

So what Contrast does there is instead of trying to have a signature and a patch for that particular CVE, and over the course of the next year, there were – I don't know, about a dozen different expression language problems in Struts and Spring. Instead of trying to issue an signature and a patch for each of those, Contrast sort of fundamentally prevents expressions from being used as exploits. So we detect the vulnerabilities. We can report them to you, but we also prevent them from being exploited. A little bit like the way that you know how DEP and ASLR work to prevent buffer overflows? [inaudible 00:35:52] whole classes of attacks.

[00:35:53] JM: By the way, I do not. I do not know. I did not know that.

[00:35:57] JW: So those are technologies that are built in to modern C and C++ compilers to prevent – DEP stands for data execution prevention, and ASLF stands for address space randomization layout. The idea here is that even if an attacker can find an improper use of gits of format string or something, that they won't be able to get their attack to be executable on the stack of heap.

So we did something similar, but for a number of different kinds of attacks. So we've got essentially a sandbox around the SQL execution engine that prevents it from being used for malicious purposes. The expression language prevention, for instance, says, "Hey, we'll detect attempts to exploit expression. Even if one sneaks through, we're going to prevent things like creating sockets and writing to files and doing system commands while you're evaluating the expression." Because that's not something that people need to do. So it's a really powerful way of defusing whole classes of attacks. That way, we can address thousands of vulnerabilities all with one technique.

But we do provide rule updates pretty frequently. We're about 160 people now. We'll be 250 by the end of the year, and we're all working really hard to cover all of those different environments and make sure that we've got expansive rule set across all of those different platforms.

[00:37:30] JM: This is a newer framing of security to me, this idea of the different classes of security problems, I guess, at the code level or at the syntactical level. But it makes a little bit of sense to me, because what you described with the way that you instrument the code with sensors, you take the code in its – What is it? I guess it's like in its machine language or after it's been – Usually when it's compiled, the C code, or usually when it's in like – It's pre-assembly, right?

[00:38:04] JW: For Java and .NET, it's bytecode.

[00:38:06] JM: Bytecode, yeah.

[00:38:06] JW: For Ruby and Python it's just straight scripts. So whatever the different platform is, but it works roughly the same way in each of those environments.

[00:38:16] JM: Oh, really? Okay. So you instrument like actual Ruby code or actual Python code.

[00:38:20] JW: Yup.

[00:38:21] JM: Okay.

[00:38:21] JW: As it loads, we focus the glass loading process again and we instrument it as it loads, yeah.

[00:38:26] JM: Interesting. So there is some way in each of these languages to kind of understand what are the risky security anti-patterns that people have implemented in each respective language.

[00:38:40] JW: That's right. Ideally, someday, we switch from just enforcing anti-patterns to actually enforcing positive rules, patterns, because it would be easier to actually have a rule that says, "Always use a parameterized query and follow these certain coding guidelines." We can enforce that kind of rule too, but out of the box, we come with a bunch of negative rules, because people require that flexibility.

[00:39:06] JM: That's a pretty cool aspiration, because then you could put that into your CICD pipeline and then you push your code and then it's like, "Okay, your code passes all the tests, but you get a yellow here, like you've got a little vulnerability here, and yeah, we've got instrumented with the sensor. But if you want to like kind of address these altogether, we can just make a GitHub issue."

[00:39:27] JW: That's really for an organization that's gotten past the flood of vulnerabilities, because right now they don't want to see the yellows because they're dealing with a pile of reds. But what we've seen is really interesting. Even large organizations – I'd give you an example of one very large technology company. They put Contrast into a division of 5,600 developers across 1,200 apps, and it took us about a year to get deployed across all of those applications, which is blazingly fast for security.

Within about six months of starting to use Contrast, most of those projects had worked off their backlog, and it was a very large backlog. They got to a state that I call the new normal, which is essentially they've worked off their library vulnerabilities, they've worked off their custom code vulnerabilities and they get to this place where they're basically clean.

Every once in a while they'd introduce a new vulnerability and then they'd fix it right away and they'd get back to clean. That's how security should work. But if you look at what we've been able to do over the last 20 years, it's really disappointing. I wrote the first version of the OWASP Top Ten in 2002, and it's still basically the same stuff in there.

I think we've learned that we're not going to be able to hack our way out of this. We're not going to train our way out. We're not going to static analysis our way out of this. We're not going to maturity model our way out of this. But I really do think that we can instrument our way out of

this. We can make dramatic changes in the way that people handle their application security at enterprise scale with instrumentation.

[00:41:03] JM: That example of an organization with – What did you say? 5,800 developers across 12 GOs?

[00:41:09] JW: Yeah, 5,600 developers across 1,200 apps.

[00:41:13] JM: Okay. Sorry. 5,600 developers across 1,200 apps that actually managed to get a lot of their security issues ironed out and get to a "new normal" as you said. That's a pretty appealing aspiration for a lot of companies. How realistic is that? Was there something about that organization that was like, "Okay, these are uniquely good developers. It's a uniquely clean organization." Maybe they were only started like 5 years ago or 8 years ago or something like that. How realistic is it or an organization, like I'm thinking of like – I don't know, I always think of like a hundred-year-old bank or a hundred-year-old insurance company, like can they have that kind of thing?

[00:41:57] JW: Yeah, they can. This organization was pretty standardized in terms of their infrastructure. So that made it really easy to deploy Contrast quickly, because we made it part of their standard environments and they just pushed it out. So we got spread very quickly. We do run into a lot of organizations that we've got all kinds of different platforms running in all kinds of different environments. There is a little bit more work to get the Contrast agent deployed into each of those projects, because it's a little bit different every time.

But it's way easier than trying to scan, because all the Contrast work is kind of upfront. It's, "Get the agent out there. Get it working. Get the data flowing," and then everything else is kind of automatic. With scanning, to do it right, you've got to run SaaS'd, and DaaS's and a software composition analysis tool and you have to do that every time you change the code. So the faster you move, the more scans you have to run, and the work just keeps piling up and piling up because every one of those scan reports has a ton of false positive. It's got to be de-duplicated and triaged. You got to map the findings to the right person to go fix the code, and all that takes time. So you don't get your vulnerabilities back immediately. You get them back weeks or months after you made the mistake. So it just doesn't encourage learning. So we think giving

instant feedback with accuracy and write to tools that people are already using is how developers want to work. So we've seen developers really like this approach to security.

[SPONSOR MESSAGE]

[00:43:41] JM: Failure is unpredictable. You don't know when your system will break, but you know it will happen. Gremlin helps you prepare for these scenarios by testing how your system responds to duress. Gremlin provides hosted chaos engineering as a service drawn from techniques pioneered at Netflix and Amazon.

Prepare your team to prevent disasters by proactively testing failure scenarios. Use Gremlin for free by going to gremlin.com/sedaily and find out how Gremlin can fit into your software development process. The first time I was at a company that dealt with an outage, I was terrified. I was at a stock trading company, and Nasdaq, which is the second largest stock exchange in the United States, froze up. This was a bizarre kind of outage, but we had to deal with it.

You will have unique outages at your company. Maybe the Nasdaw will freeze up and you'll have to deal with it, and the best way to harden your infrastructure to be ready for those stressful events is to put stress on your infrastructure today. Go to gremlin.com/sedaily and learn about the different ways that Gremlin can chaos test your infrastructure.

Max out CPU, black hole, or slow down network traffic to a dependency. Terminate processes and hosts. Each of these shows your system reacts, allowing you to harden things before production incident. Check out Gremlin and use it for free by going to gremlin.com/sedaily.

Thanks to Gremlin for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:45:33] JM: This show covers elements of business as well software engineer. We also cover some management issues, and you are pretty advanced in the state of the company, and we have not covered company development of a security company with anybody who has gone

from the introductory startup stage of a security company to your point. You've raised $119 million at least, from what I know. The latest is series D. I'd love to know how the engineering strategy has changed throughout the stages of company maturity and stages of funding.

[00:46:16] JW: Yeah. We've obviously grown a lot. We started with just a handful of people at my cofounder's mom's attic, and we made a really good hire early. Our VP of engineering is a guy named Steve Feldman who was the VP of engineering at Blackboard. He's one of the early employees there and grew that to be a massive devops shops, and he was great.

When he joined us, he started putting in devops practices. He didn't just run in and try and change everything. He started introducing practices. So we sort of got organized around two weeks [inaudible 00:46:52] at first and then overtime we accelerated that. Now we're at like 7 or 10 deploys a day. He introduced things like show and tell and we got really serious about tracking things in GitHub.

I think one of the major evolutions was when we broke out into teams for each of the agents and our team server, sort of structurally then we had to build a management structure at each of those teams. We're still growing really fast, so that's continuing to evolve. But we've been devops from the beginning. We really focus on trying to automate all our testing and our whole pipeline. So maybe that gives you a sense of how we work.

[00:47:33] JM: When did you stop coding?

[00:47:35] JW: That's a great question. I did a lot of the work on the early product building the agent and figuring out how to do instrumentation for this along with my cofounder, Arshan Dabirsiaghi, and so up through the series B, maybe, I was pretty involved with coding and adding features to the agent. But I'm always been sort of an agent guy. There's quite a lot of work in our team server as well, but I haven't done any of that.

I did jump in over Christmas a year ago, I helped to work on a version 2 of our RASP engine. We call it REP, or runtime exploit prevention, and that's where I was getting into some of the different layers of protection that we can weave into the application. So we really went from just a basic RASP to RASPNG or what we call REP, which is not trying to block attacks, but really

trying to prevent the exploitation of whole classes of attack. So I got to do some coding there and that was great. I really enjoyed it. I miss it a lot. But a lot of my life now is sort of evangelizing and meeting with customers and partners and so on.

[00:48:43] JM: For a lot of entrepreneurial software engineers out there listening, they might not really consider security as an area that they can start a business in. To my mind, when I think of security, like selling a security software product, I just imagine myself having to get up and like put on a suit and tie and like go into an office somewhere, and it's like sales, sales of security. I mean, sales and software is hard enough or complicated enough. I really want some kind of self-service thing. I want to avoid the sales model altogether. What was your approach to closing the first sale, and would you caution software engineers against starting a security company where you really have to sell into these kinds of enterprises?

[00:49:35] JW: I would strongly recommend against it. It's really hard. But maybe that's just from where I'm sitting now. So this is interesting I think, because a lot of startups start selling to small business and then they try to move into the enterprise space. We did that the other way around, because we started working on Contrast at my consulting company, Aspect Security, and we spent several years there working on Contrast, and our customer's aspect were all very large financial institutions, like Federal Reserve and Citi and J.P. Morgan and so on.

So our first place, the easy place for us to go was to talk to these giant customers. So we got very early feedback on the enterprise features that we need. So we had to add a whole bunch of stuff really early, like access control, and logging, and we had to get SOC 2 Type 2 compliance really early. So we did all these really hard stuff early. That actually ended paying off later. But that was tough, because they wouldn't buy anything before we had all that stuff.

[00:50:41] JM: You brought in an external CEO. That is something that was kind of, I guess, not a shibboleth, but people didn't do that, or I think entrepreneurs don't think of doing that usually. I mean, there are some amazing historic exceptions to that rule, like the Eric Schmidt example, the HarshiCorp team has brought in an external CEO. It's going really well there. When did you bring out an external CEO and why?

[00:51:11] JW: Yeah. So I was CEO of my consulting company, and we were a decent size. We grew at about 60 people. It eventually sold to Ernst & Young a few years ago. So that was a – I had some CEO experience, and I looked at this and I – Well, I was a little arrogant. I mean, I really thought that this technology had a lot of promise. So I thought, "You know what? I've never – I don't know anything about building a product company. I know how to build a consulting company." I think you got to look at yourself and say, "Do you really have the skills to do this job?"

When I look at the Java, the venture-backed technology company, is it's fundraising, it's working on MNA stuff. It's dealing with investors and partners and – I looked at that and I just said, "You know what? I've never done any of that stuff before. I'm going to find somebody who really knows how to do that and I'm going to stick to what I'm really good at, which is security and software, and I'll take the CTO role." I was very happy to do that and I think that was the right thing for my company.

[00:52:18] JM: What stage was it? What funding stage or what maturity in the company?

[00:52:21] JW: Yeah. So we did kind of an angel round when I was still at Aspect, but our A rounds was when we brought in our first CEO, and it turns out unfortunately that there's the job of building a company from sort of 0 to 10 or 20 million in revenue is a little different than building it from 10 to 100. So you may find yourself switching CEOs, and we did that to our B round. Again, that was a great choice. We brought in Allan Naumann, who's our current CEO, and he's been absolutely fantastic. He really is helping us grow to that next level.

[00:52:58] JM: I heard a great quote this morning. Actually, I was listening to The Twenty Minute VC, which is a favorite podcast of mine, and one of the guest was the CEO of Carta, and he was talking about why executives sometimes churn out pretty quickly. This quote I really like is, "People often grow linearly, but companies grow super linearly, like exponentially."

[00:53:21] JW: Oh, yeah. That's interesting.

[00:53:22] JM: So like that's kind of a mismatch there. But how do you align incentives correctly? When you bring in an external CEO, like first at like the series A you – I mean, you

can't exactly give them founder level equity, but you want to give them a reasonable amount of equity so they have skin in the game. Then again, the series B, like you kind of have the same problem. How do you align incentives?

[00:53:43] JW: Yeah. Well, you can vest that equity so that it takes time and they have to earn out and you can set targets in place to kind of make sure that the company is going the way that you want in order for them to earn the equity that they get. Yeah, you want someone really good, they're going to have to feel like they've got some skin in the game. So you want to give them the right amount of equity, and it's worth it. If you really want to take a shot at being a huge company and going public someday, then you've got to get the right people in place.

[00:54:13] JM: I mean, the reason I ask is I'm just like I like the idea, the external CEO. I can imagine myself like in a situation someday where I would want an external CEO of a company that either I am a part of or that I'm running myself, and I can imagine that's just being really hard to calibrate and like I can imagine it being something that's like crucial to the company and also there's like maybe not that much information out there about it.

[00:54:40] JW: Yeah. Our search for our current CEO was a lot. I mean, I think we talked to probably 20 different CEOs and we looked for somebody who really left of the page. Your investors can be super helpful there, because they're often really plugged in to the community of CEOs.

[00:54:57] JM: And incentives.

[00:54:57] JW: That's right. They can also kind of – Yeah, they can kind of help you understand what the market is for a good CEO and make sure that the incentives are aligned with their background and skill level and so on.

[00:55:10] JM: Okay. We're up against time. Let's wrap up with a totally unrelated question to external CEOs. How big of a problem for security is cryptocurrency related attacks?

[00:55:21] JW: Yeah. So it's a completely different kind of security than what we do, I mean, for the most part, because mostly the kinds of machines that people compromise are desktop

machines and cloud servers an stuff, but I think it is a very interesting attack. People figured out a way to use hijacked resources to make some money.

The part of it that interests me is if somebody can knock over a web server, an app server, and turn it into a cryptocurrency mining thing or a ransomware thing or something. We haven't seen too many of those attacks, but it's certainly possible. Like that Equifax attack, instead of stealing that data, somebody could have easily knocked over their app server and turned it into a crypto coin mining farm or a ransomware kind of thing where they took Equifax hostage. So those kinds of things are pretty scary scenarios.

[00:56:13] JM: Okay. Well, Jeff Williams, thank you for coming on the show. It's been really fun talking to you about security and business building.

[00:56:19] JW: Thanks, Jeff. Appreciate the time.

[END OF INTERVIEW]

[00:56:25] 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 plug-ins. Use the value stream map to visualize your end-to-end workflow, and 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 longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]