Micro Focus Reimagining Cyber Episode 41 Dan Lorenc

Tue, Oct 04, 2022 10:11AM • 24:57

SUMMARY KEYWORDS

folks, software, certificates, SLSA, supply chain, vendors, people, typo, dan, open source, reimagining, publish, company, problem, build, couple, solarwinds, cyber, attack, helping

SPEAKERS

Rob Aragao, Stan Wisseman, Dan Lorenc

Dan Lorenc 00:00
Especially in the federal government, and a lot of these agencies, a lot of the software does still run on prem. That's a pretty big problem. And Log4J was a great example of that, where anybody running anything with Java, had to go around and manually email their vendors to figure out if Log4J was in there and what version and if they were affected, and then what to do about it. They had no idea. Folks didn't even know what language it was in and there's some hilarious stories on Twitter about you know, that the maintainer of Curl was getting spammed with emails from vendors demanding he responded with another Curl headlock for J inside of it. It's not even written in the right programming language for that, but you know, for sure, just panicking. You know, everyone that could because I had no idea.

Rob Aragao 00:43
Welcome to the Reimagining Cyber podcast where we share short to the point perspectives on the cyber landscape. It's all about engaging yet casual conversations and what organizations are doing to reimagine their cyber programs, while ensuring their business objectives are top priority. With my co host, Stan Wisseman, Head of Security Strategist, I'm Rob Aragao, Chief Security Strategist. And this is Reimagining Cyber. So Stan, who do we have joining us for this episode?

Stan Wisseman 01:09
Rob. Our guest today is Dan Lorenc. Dan is the founder and CEO Chainguard, Inc. and Dan is also a former Googler, who has been deeply involved with a number of projects focused on mitigating software supply chain security risks, like the six door project and the supply chain levels of software artifacts or better known as SLSA. Dan has been involved with the Cloud Native Computing Foundation, he chaired the continuous delivery foundation technical oversight committee, and sits on the governing board of the Technical Advisory Committee for the Open Source Security Foundation. Dan, you don't have any time to sleep, man, you are involved in so much. Is there anything else you want to share with our listeners about your background?

Dan Lorenc 01:52

I think I forgot about some of those until you said, No, thank you. I think you covered it pretty well there.

Stan Wisseman 01:59
It's great to see how active you are, I'm not that I don't have that much energy anymore. You know, as you are everybody as well, where we've had a number of high profile incidents associated with supply chain software. And it certainly heighten that awareness of the need to protect against threats targeting the intentional injection of malicious code into software dependencies. And I'd like to start with asking you about your work on the SLSA framework and my understanding. And granted, I have not had a deep level of involvement with SLSA, but it's an end to end framework, trying to help developers ensure that the integrity of their artifacts around software from source code to the build process and other mechanisms in the development of software, have integrity. How did SLSA get started at Google? And you know, why do you think this approach of mitigating intentional threats to the software supply chain can work?

Dan Lorenc 03:04
Sure, yeah. So the SLSA project or supply chain levels, or software artifacts is what that stands for. We started at Google with some of my colleagues, it's not brand new, none of the ideas in there are really new, it's just kind of set up in this incremental way that we put some time into thinking about and then published. But it's really, you know, if you read it, there's a bunch of levels and a bunch of requirements and, you know, a bunch of kind of random seeming things that all tie together. But, it really ties back to one core principle, and then thinking through how you apply that principle to modern software development. And that principle is basically two party review, or multi party review, for anything that can affect you know, the end result, the end piece of software in the end binary, the end deliverable, the end container, anything like that. And you can write that one principle down, but then when you start thinking about it, you know, that applies to the source code coming in at first and cool, you know, most people do source code review today. And also applies to the tools used, right? If you compromise a compiler, or you know, one person gets to choose, the compiler that gets used, doesn't matter how many folks reviewed the code, you know that that compiler is the one in charge of producing the executable? Can you apply that to the compiler, and you have to kind of declare your build environment, and you have to have multiple people review that too. And then you have to operate it in a secure manner. So one person can't just bypass all those controls. And that one principle extrapolated to you know, package managers and CI systems, and everything is kind of how you end up at all the requirements. And the SLSA framework,

Stan Wisseman 04:30
You think it makes an impact? I mean, as far as actually implementing that kind of approach.

Dan Lorenc 04:34
Yeah, I think it's the only way to make an impact. You know, and this kind of goes all the way back to kind of the birth of software supply chain security or the kind of birth of the field which was Ken Thompson's famous paper from 1984 "Reflections on Trusting Trust," where he basically points out you can't trust computers, you can't trust tools, you have to trust people. And people are hard to trust, especially in open source and you can never trust anybody completely. So the next best thing is make sure that you're having multiple people check each other's work. And you see a huge difference from unilateral access, one person be able to do anything on their own, to two party review, right, that's the

biggest jump you can make, you can go farther, right, you can stretch it to three, you could stretch it to four people, but start to get incremental benefits past that, you know, the likelihood of compromising one account through phishing or something like that is way higher than two accounts or something like that, that happened to be cooperating on a team or next to each other. So it's really about nothing is perfect. The only real way to do this is by increasing that number of people that have to be compromised or hacked or something from one to two to three, something like that. And then applying that recursively.

Rob Aragao 05:45
So Dan, you know, we continue to see the attacks on software artifacts, right impacting supply chains, we've been to kind of touching upon that a little bit. But outside of the main headlines, you know, kind of solar winds, everyone continuously looks back at, you know, what are some of the other examples that you see out there, that's been prevalent.

Dan Lorenc 06:01
Yeah, those are kind of SolarWinds is kind of the big kickoff, right, I've been worrying about this space for a while back to like 2016, or 2017. And for those first couple years, nobody really cared, nobody was really too worried about it. We saw for focusing on just integrity kind of SolarWinds is where it really did start to jump off. But software supply chain security in general is a couple of different problems rolled into one, right, there's that integrity, one of folks getting access to build machines, compromising artifacts, that kind of thing. There's a separate one, though, which is the quality problem. And so we've seen some waves of that in the past. I mean, the huge example there is Log4Shell, Log4J. It was on the one year anniversary of SolarWinds, almost to the day, I think. And you know, those two together get lumped together into the supply chain security category, but they're very, very different, right one you can't even call Log4J an attack or Log4Shell an attack. It was just a bug that was unintentional sitting around for a decade. And then folks tried to attack it when they found out about it. But very different than the SolarWind style APT nation state targets, that company actually does attack them and insert some malware. Those are the big two, at least in the past couple of years. But we see more and more. And I think it's a reflection somewhat on us getting decent at other forms of security. Phishing is getting harder now that MFA is you know, becoming prevalent. We're using HTTPS everywhere finally. If you remember using the internet, you know, 5-10 years ago, you know, a lot of websites still didn't have certificates that were valid. Folks, we're sending passwords in plain text to other banks at Starbucks over the WiFi kind of thing. So you know, we've gotten good enough at hardening a lot of these other attack points, and the supply chain is becoming the most attractive way for attackers now.

Stan Wisseman 07:38
One of the attacks I hear about is typo squatting. Can you expand for our listeners, what that means?

Dan Lorenc 07:46
Sure, yeah, typo squatting. That's a really sinister one, because it's almost more of a form of social engineering. And like I said, multi party review people are the answer. People are also the problem, right? We are the weakest link. Type of squatting is a tactic. It's been used in all sorts of areas. But it keeps happening in open source and in software supply chains, where an attacker will take a popular package, something that you know, some library that a lot of folks use, and then kind of push a version with a slight change in the name, whether it's a typo, or you know, even something more innocuous,

like inserting a hyphen in the middle or something when you don't know exactly how the word is supposed to be spelled. And then try to trick folks into downloading that other version. You can see it with swapping out like a Unicode character. So you know, even if you're reading it, you can't tell it's a typo. It's actually just a different character or something in there. Or you see, you know, folks just change or reorder the words or something like that. During my time at Google, we saw it happen all the time with libraries we publish on like Ruby gems, or pi pi, or something like that, where it'd be called something like GCP hyphen, client hyphen library or something. And then somebody would immediately publish like Google Cloud hyphen client library, and like, it's not a typo. And if you're just an end user, how are you supposed to know which the credible one is. It's really just kind of an attack on naming being hard and imprecise and more social engineering to trick folks into installing something like that? So there's no real known solution to that really today. It keeps happening.

Stan Wisseman 09:09
You mentioned the different levels in SLSA. I mean, what is the difference between the different levels? And how do you see organizations pursuing a particular level? And do they try to go all the way to the right to like a level four? Or do they shoot for a moderate, you know, two or three?

Dan Lorenc 09:30
Yeah, it really depends. We broke those levels down for a couple reasons. SLSA was actually based on an internal framework we had at Google that's been written about on a couple of white papers. But um, it was kind of the same principles but applied inside of Google's environment. And Google's environment was a lot different. So we couldn't just kind of copy paste and apply those same things outside. But um, Google environment is based on a single giant mono repo. A lot of folks have heard about all the code is in one massive repository for the entire company. And it uses a version of the, it's open source, it's called Basil the compiler, but internally, it's called Blaze, where everything is built from source and kind of one step using that tool. So there's no intermediate packages that get checked out or fetched anywhere, everything goes straight from source to production. And so those same principles are a little bit easier to apply in that environment. And then if you have, you know, intermediate packages getting published by one person and downloaded and then turned into another binary, and then that gets turned into a container or something by a third person, that's where it started. And, you know, that was also set up that same way four different levels. And they're kind of set up that way for teams to plan and folks to be able to take an incremental approach and recognize the fact that not everyone needs to be at the highest level. So it's a great tool for CISOs and technical leaders to be able to set goals for their team and have it decoupled enough where the team gets to figure out the how, and they just get to set the what like, you know, next year 90% of our critical loads have to be at level three, and folks have 12 months to then go figure out what the best build system they want to use is and how they want to get to that level on their own, without kind of meeting all of those decisions for folks. So I think, you know, right now, we are seeing a lot we're seeing folks grab it. And they love that concept of I can just pick a number and a target and let folks you know, figure out how to get there on their own. And they will see you go up over time. I think level four is a pretty big jump today, if you actually look at the requirements from level three, I think that's where the first wall really gets it. To put it simply in words like level one is basically use a build system. That's kind of how I think about it don't do builds on people's laptops use like a build system. Level two is store the logs from that somewhere. So you can look them up later, if you want to, you can still get and use pretty much any build system for that though. Level three now starts to be like user one can configure it really well make sure you're

operating it securely. And level four is most built systems out there today don't even support this stuff. So it's gonna take a while before folks can get through it unless they want to roll their own. It's, you know, hermetic builds with no network access and reproducibility at every step, that kind of thing. So unless you're deploying to a really sensitive environment, most folks aren't going to need to be at level four today.

Rob Aragao 12:02
Yeah, it sounds to be kind of not the extreme, but it'd be nice to get there, obviously. But you know, some of the SLSA source code requirements are around verified history and provenance, right? So this kind of takes us down to the path of project six door, digital signature verifies kind of who signed the code, code has not been tampered with. We think about those six doors approach, what's different, versus the kind of common code signing practices in the taken place by vendors in the open source community?

Dan Lorenc 12:32
Yeah, great question. Six door project got kicked off in mid 2020. Sometime around then, to try to figure out why digital signatures weren't more widely adopted across open source, there have been options. For decades, PGP has been around forever. You know, PGP is set up with this web of trust model where you get to meet folks in person and exchange keys and sign each other's keys after, it's complicated. Yeah, and I think my theory, there's never really reached a critical mass, because folks are bad at key management, you know, there's a couple of fundamental problems with users today. And you know, password management, key management is another one, the PGP model can work, if you're pretty confident folks are gonna have a key for decades, that kind of time, and I can't keep my phone for decades, you know, I lose stuff constantly. And I think you know, because of that folks lose the keys too much to really build up that critical mass of that web of trust. So never really took off outside of kind of hardcore circles where folks care more about that kind of thing. And then there's also like the kind of corporate model certificate authority world where you send a fax of your incorporation letter and business address, on letterhead to a company and pay them a couple 100 bucks, and they give you a certificate, that's good for a couple years, and you get to use that to sign your code. And that's more widely adopted, but really only in like, the walled garden ecosystems of like, you know, Windows drivers or the Apple App Store, that kind of thing, where they can control it, they can issue certificates, and they can kind of force people onto these models. So looking at all of that, we saw a bunch of parallels to the Let's Encrypt model. For those familiar with Let's Encrypt, they were kind of a nonprofit, public Betterment group that got started up to help do certificates for the internet. So we talked about touch on a little bit earlier. But you know, five or 10 years ago, it was pretty hard to get a certificate for a website, you had to do something similar. You had to find a CA pay them some money, they email you back and forth, and you'd get a dot PEM file, and you'd have to copy it into your Apache config and figure out how to get it into the right place to get the domain to serve. And big companies did it and you know, even then, they're still embarrassing, like forgetting corinium certificates and sites going down on certain days and stuff. And Let's Encrypt really wanted to figure out how to do that for the entire Internet. So they built some new protocols to automate that process. They work with the browser's to get their trusted and showed that if it's automated, and you know, the certificates are only good for a couple of weeks and people are constantly refreshing them that it's actually a more secure world than the old one is manual verification and credit cards and stuff. And it worked and you know, over a couple of years I think TLS adoption went from like 50% to like 98% or so across the Internet

was massive, by doing it for free and automating it. So we basically copied that playbook. But for open source code signing, we built a free certificate authority that you can get certificates from, they're only good for a couple of minutes. So you don't have to worry about keeping them or losing them or leaking them or anything like that. And it's all automated. So every time you want to get one, you verify your identity. And the easy part was these protocols already existed, we didn't have to make a new one, like Let's Encrypt had to do, it's called Open ID connect. And you know, the browser window just pops open and you click login with Google and you verify your email address each time. So you can issue those for free and in seconds, and you don't have to manage keys, you don't have to think about it. But one thing folks can hold on to over long periods of time is their email address, right? You know, you'd be kind of lost without one of those without that same one. MFA, all that stuff has helped protect that over time. So we base everything off of that email address, that also happens to be the way folks identify themselves in open source. It's great because it can be as anonymous or pseudo. I can never pronounce the word pseudonymous pseudonymous, you know, as you want it to be, it can be your real name and a real company, or it can be an alias, and you got multiple of them, depending on which communities you're working in at folks do that. And it really just kind of allows you to build up a stable identity over time and make sure the code ties back to that.

Stan Wisseman 16:19
And what's the adoption rate? I mean, as far as any open source projects, I mean, are you seeing an uptick?

Dan Lorenc 16:24
Yeah, it's been massive, like this is only been around a year and a half, two years, something like that. And then we're seeing kind of both the huge large projects adopted, because they were struggling with stuff like Kubernetes, you know, the fastest projects, now signs all of their releases with it, a couple of weeks ago, Python, C Python interpreter started doing this for all their releases. And also, we're seeing kind of the long tail support. So a lot of these package managers didn't support any signing at all. Some had, like GPG support that wasn't really used, but some just didn't support it, because there wasn't enough demand. And then we're seeing like GitHub just announced NPM, the node package manager will support safestore. Python support for pi pi as well, Ruby gems, Maven Central, all of these open source package managers are now allowing it for their developers to use.

Stan Wisseman 17:09
Well, I'm going to switch topics, you know, we had an episode last year, right, Rob, after the Log4J incident. Steve Springett. And, you know, we were one of the topics, obviously, was the Log4J incident. And this topic around software supply chain. And we got to the topic of software bill of materials, and his work with AWASP, Cyclone dx. And you know that they have done a lot of work around creating this lightweight, quote, unquote, SBOM kind of framework, or format. And, you know, the objective of an SBOM, if folks haven't seen that heard that episode is really to try to enable you to understand the components that make up a software package and be able to hopefully, with that greater awareness, respond faster if there's an incident, you know, first off and understand your risk inherent to the components that might have vulnerabilities within them. But also, if there is some kind of incentive, like for Log4J, you can respond faster. But Steve made an observation that look as pawns alone, that really is inadequate. Do you agree with him on that? And if so, what else do you think needs to be done?

Dan Lorenc 18:29
Completely. Yeah, I agree completely there. You know, I started out as an SBOM skeptic, I've come around a little bit, too, you know, they do have a couple of use cases where there

Stan Wisseman 18:37
It does seem to be a religion, you know, as far as whose side you're on, and all that.

Dan Lorenc 18:42
Yeah, I've come around, I think, you know, there have been a bunch of docs and use cases and stuff published on, you know, things that SBOM can help with. And, you know, I've settled on a, you know, two core ones, I think, you know, do make sense. And we are seeing some usage for them. And, you know, the regulations and stuff asking for them are helping there. The big one is the inventory management piece and transparency that way. And I think that's probably what Steve was referring to. And that's, you know, when you buy a traditional piece of box shrink wrap software, vendor gives it to you, you install it on premise on your servers on your machines or something like that. Before SBOMs, before this whole world of transparency, you were kind of just at the mercy of the vendor in case something went wrong, you had to trust that they would tell you if something was wrong. And you know, for the most part, vendors are well intentioned, you know, they try to do that they publish, you know, disclosures, but it's all kind of on their website, you might not know where to check, they might do notifications, but it's sort of like buying a physical product, like unless you fill out all those little registration cards and keep your contact information up to date. They might not get the email to you because you never told them your email, or they forgot about it or something like that. They don't know that you're still running a seven year old version, that kind of thing. And so especially in the federal government, and a lot of these agencies, a lot of the software does still run on prem. That's a pretty big problem. And Log4J was a great example of that where anybody running anything with Java had to go around and manually email their vendors to figure out if Log4J was in there and what version and if they were affected, and then what to do about it, they had no idea. Folks didn't even know what language it was in. And there's some hilarious stories on Twitter of like, you know, the maintainer of curl was getting spammed with emails from vendors demanding he responded on whether curl had Log4J inside of it. Not even written in the right programming language for that. But you know, folks are just panicking emailing everyone they could because they had no idea. So I think in that scenario, SBOM is pretty valuable, it starts to get stretched a little bit too late, you know, folks asking for them for SAS products as well. And, you know, transparency is always good, you know, tell me the SBOM for gmail.com or something like that, sure, I want more data, but I'm a little more skeptical there like, what are you gonna do with that data, you can't go patch that server, you can't turn it off, you know, the vendors already on the hook for that. So it's a little less useful in that case, in my opinion. The second case, where it's useful, though, is purchasing decisions. We've heard that one a bunch, if you're a company, and you're trying to evaluate two different vendors, or two different pieces of software, and you're trying to get an understanding of the security posture of both and you want to use that to inform your purchasing decision, you got to ask around, and you know, if somebody gives you one, and it's written in a 30 year old, unsafe programming language with out of date dependencies, or known vulnerabilities or something like that, and the other one hands you a nice slim one and modern programming language with garbage collection and up to date dependencies and stuff like that. And that's a pretty big signal, you don't need to get down to every line and review that kind of thing, you can just kind of tell that at a high level. And you do see that one come up too. And I think that's something a lot of folks in the

government are excited about getting that transparency. But you know, SBOM is pretty good at that known vulnerability, asset inventory problem, but it doesn't solve like, you know, the integrity piece that we started with.

Stan Wisseman 21:54
Right, it didn't help you with the intentional injection of malicious code into your software development processes.

Dan Lorenc 22:02
I think that's why I was a little skeptical at first, you know, much of the debate and discussion around SBOM came up right after the SolarWinds attack and like folks weren't expecting line 37 In this SBOM to contain like, malicious DLL dot package from this APT kind of provenance information in there. Wouldn't have helped at all with something like that. But you know, it is part of the larger problem.

Rob Aragao 22:26
So Dan, kind of pulling it all together, right? Are you seeing, you're right there in the driver's seat, you're involved in a lot of these different projects been going on for quite some time that you've invested your time and energy into. Do you feel we're making the right progress as relates to software supply chain risks?

Dan Lorenc 22:40
I think progress is finally being made. Yeah, I think the regulations are really helping each month, each quarter each week, sometimes there's new announcements coming out of the federal government. And you know, they don't write a lot of software. So it's a little bit risky, and having them write regulations on how software is supposed to be written. But they are one of the largest purchasers of it in the world. And so they have a lot more information and your experience buying software than most folks do. And so kind of combining those two, having the industry consortium is helping inform a lot of the stuff on best practices. And then having, you know, the government being willing to put the money where their mouth is, and say, We're not going to buy software, unless it's been produced in a secure way, I think is driving a lot of change here. It's a classic type of, you know, negative externality style, like long tail risk, where you know, there's 1000s of companies, and probably only two or three or 10 are actually going to get hit, and hopefully their cyber insurance is going to cover that, why should I change the way I do stuff. But until somebody comes along and actually puts regulations in place, it's gonna be hard to get the industry to take it seriously. But thankfully, that's happening, right. It's been a year and a half since the original executive order, from the Biden administration and stuff has been moving incredibly quickly for government pace.

Rob Aragao 23:51
Dan, thank you for coming on. We really appreciate the insights that you shared, I think one of the main things we can take away from what's kind of been almost a catalyst. It's unfortunate, but reality is what happened with SolarWinds turning back the clock, right. Then here comes as you just mentioned, the executive order makes things happen, the government gets involved and, you know, yeah, they are actually helping progress this forward. And your insights also at the end that you truly can be in the driver's seat, see the actual differences and steps being taken the right direction. So thank you very much for coming on and sharing your insights. We really appreciate that.

Dan Lorenc 24:22
Thanks for having me on.

Stan Wisseman 24:24
Thanks, Dan.

Rob Aragao 24:26
Thanks for listening to the Reimagining Cyber podcast. We hope you enjoyed this episode. If you would like to have us cover a specific topic of interest, feel free to reach out to us and you can find out how in the show notes. And don't forget to subscribe. This podcast was brought to you by CyberRes, a Micro Focus line of business where our mission is to deliver cyber resilience by engaging people process and technology to protect, detect and evolve