FEDERAL TECH PODCAST

Ep. 32 Software Supply Chains and Federal Technology

Welcome to the federal tech podcast. My name is John Gilroy and I will be your moderator. Our guest today is Dr. Steven McGill, Vice President product innovation at Sonatype. Steven, how are you? I'm good. Thank you. He's not a regular doctor. He's a PhD in computer science. So we got to ask him computer science questions, not questions about my sore back or my elbow than certain. And so today the question is, what's his whole idea about the software supply chain? And what's the vulnerability? And specifically what kind of vulnerabilities does the federal government have? And, and I think anyone listening to this podcast knows that this isn't, you know, a brand new topic. It's been talked about a lot, in fact, luminaries, I love me the word luminaries, luminaries have chipped in on this. In fact, one of my heroes is a guy named retired brigadier general Greg touhill. And, and recently, he commented that, you know, if you, you're worried about every single line of coding that you ever use in the future, you're not going to do anything, the Security's going to be out of hand. And so Steven is going to join us and, and maybe give us some insight as to how the federal government can get a handle on the code that they're using. But before we begin, Steven, tell us about your background, please. Sure,

02:32
yeah. So I got my start in research, doing my PhD work in code analysis, analyzing software for security issues, performance issues, and so forth, basically static analysis, I continued to do academic work for a while and then sort of got more and more interested in how to take these tools and techniques and really get them in the hands of developers and have an impact on on software development and improve security practices. So I started moving more and more towards product, I founded a company called Muse Dev, to produce tools to help developers analyze their software as part of the development process. We got acquired by Sona type a couple of years ago. So that's how I ended up here. Although I've been collaborating with the folks at Sona type on the state of software supply chain report for quite a while. This is an annual report that we do, it really takes like a deep dive into the software supply chain and risk management there. And so that's given me the opportunity to do research on you know, what's effective when it comes to secure software development practices, which has been really enlightening.

03:36
So just as people listening don't think this is some little arcane, old topic that PhDs computer science debate at a coffee shop after class or something. No, no, here's the figure that comes up here. You know, attacks on software supply chain have grown by an average of 742% each year since 2019. So something's happened. And Steve and I, yeah,

03:57
yeah. And what's crazy is that growth is in sort of a new style of attack where, you know, what we're used to, in software supply chain attacks is, you know, there's some vulnerability and in software, it's been sitting there for

a while, but nobody's really noticed it. And then some security researcher discovers it or, you know, worse, you know, some hacker discovers this issue in software and starts exploiting it, right. But more and more attackers are trying to create their own opportunities and introduce new errors into the software by gaining positions of trust with the development team, contributing code, and then eventually contributing malicious code that they can take advantage of, or there's something called typo squatting, where they publish a package that looks very similar to another package, a commonly used package. And so there might be, you know, a typo in it, or there might be a dash where there wasn't a where there was an underscore before, something like that, and sort of trick developers into using this malicious version of the software. So this this sort of new style attack has been just growing explosively.

04:58
I did some research read some observations about software scanning from the Navy and the Air Force in different groups. And is this one quote? I mean, this makes perfect sense doesn't it says, hey, you know, we're not in the business of building software source code scanners. Now the Navy is in the business of floating ships and destroying enemies. So I mean, you have to rely on somebody or some, some kind of an organization that can scan the software for you. And it has to be done. Sounds like continuously.

05:26
Yeah, that's right. That's right. And, you know, ideally, at various points in development, right. So you want to be scanning the software as part of the build and test process, so that you're catching things as early as possible. And then you want to also be scanning those artifacts that are eventually produced, you know, the things that you're going to be running the containers, the images, the JAR files, that you're pushing out into production. Because, you know, it's surprising sometimes what happens between the developer writing their code, and then everything getting packaged and pushed to production, a lot of code can get pulled in that you don't even realize.

05:58
And it's not intentional. I mean, let's paint a picture here, you've got an agency that has got limited resources, few people, maybe your CIO hasn't been there, and you have new deadlines, new pressures to come something but the certain six months or eight months, wherever the OMB wants you to do it in are the executive officer, the president pushing you and pushing you pushing you, I'm not saying they're cutting corners, but they may not compel us every single best practice in the world, would they?

06:25
Yeah, well, that's a yeah, there's a number of things that get in the way of, sort of trying to be efficient while while coding securely. And so, you know, there's, there's things like, often certification and sort of regulatory audit burdens, you know, those can introduce roadblocks, where, you know, you can't maybe move as quickly from a development standpoint as you'd like to, it's, there's also the fact that, you know, the sort of maintenance tasks that keep you secure when it comes to your software supply chain, things like updating to the latest versions of libraries that you're using, those can those can involve real engineering work, you know, sometimes it's, it's a simple change, but sometimes you make that change, and it breaks your build, you know, it causes unexpected behavior. And now, you have an engineer having to chase down, you know, new bugs

just just due to updating that that dependency. And so, you know, it's not an easy problem. It's no, it's no surprise, really, that, you know, people are falling behind when it comes to security.

07:21
At the beginning of COVID, I had the CTO or from patent trade USPTO. And, and he said, Hey, we've been remote for a while went pretty easy, and was I guess, was easy transition for them. And it's interesting that they're one of the organizations that may have had problems with some of this embedded code, aren't

07:39
they? Yeah, yeah. I mean, anytime, you know, you're, you're moving remote. Like, it introduces new challenges. So, you know, things that like, one, another area that we've seen sort of an increase in attacks, and and the solar winds attack is a good example of this, right? We've seen new style attacks, like on this software development process itself, right. So attackers getting software into the development environments, onto developers machines. And then and then exploiting that, that foothold that they have. And that becomes much easier when you're in a remote work environment, right. Whereas your, your build server and your development machines, and everything might have been, you know, on the same network and behind a firewall, and sort of in a controlled environment when you are on site. Now, you know, you've got developers coding on their laptops from home, there's laptops, or you know, going all sorts of places, coffee shops, airports, etc. You know, and that that opens up opportunity for attackers.

08:35
So while brilliant software developers, we use a phrase like transitive dependencies, and you know, that's pretty impressive. All it means is that one thing relies on the other, like building some blocks. So we know there are transitive dependencies. So so how can a federal agency be aware of those?

08:53
Yeah. So this is a really important thing to pay attention to, when you're looking at tools in this space, and sort of deciding what, what processes you should have for scanning your supply chain. You know, there's, there's this layer of direct dependencies. So these are, you know, you're writing your application, and you're like, I need an HTML parser. So you go grab a library that does that, you know, and I need to interact with this Amazon API. So you pull down, you know, the package for interacting with that API. Those are your direct dependencies, but those packages themselves have their own dependencies, often, you know, the HTML parsing library might have a logging library that it's using, you know, and so forth. And so you're actually picking up much more than just a single software package when you add a dependency. And you know, the these trees get quite deep, you know, when you think about the different levels of dependencies that can go. The average is I think we found the average was 5.7 layers deep. This is for open source projects. It's probably even deeper for enterprise projects. And, and then all told when you look at the sum total of dependencies you're pulling in the average enterprise application has 148 dependencies included, that's for Java Java code. So it's a huge number of dependencies and you want to make sure that whatever tools you're using are scanning that full dependency tree, there's actually tools out there that only look at the direct dependencies and are doing that deeper dive. And it's particularly important because that's where most of the risk comes

from. So if you look at vulnerabilities, you know, this, this application is vulnerable. Where did the vulnerability come from? six out of seven times that vulnerability is from a transitive dependency, not a direct dependency.

10:33
I taught English for a year, I keep thinking of transitive and intransitive verbs. And I,

10:38
yeah, for a long science terms a little different, but

10:41
what their language is, I guess, more or less. Yeah. So song type has something called a some type safety reading. So what does that mean for my listeners?

10:49
Yeah, so this is a new thing we developed as part of this year software supply chain research. And it's been really exciting. Because we've, for a number of years now, we've looked at, you know, metrics in the space like security and safety metrics for projects, what what can we say about the health of this projects development process, the likelihood that this project will be well maintained, will, you know, have a low number of bugs won't introduce breaking changes into your code. And so we've we've explored it, we've evaluated metrics that are out there, we've developed our own metrics, but we've never sort of made them publicly available in a way that the community can benefit from, and and can help improve. And so we've done that this year with the safety rating, it incorporates MTU, which it stands for mean time to update, which is a metric we developed back in 2019. In that year's report, and this is basically just a measure of how good is a project at keeping its dependencies up to date? So how good is it in managing it supply chain? Because like I said, a lot of risk comes in through those transitive dependencies, those dependencies of dependencies and so forth. And so you want you want a project to be good at keeping that up to date, so that you know, you're not getting that risk sort of flowing down through all the dependency chains. So that's one component of it. The other component is something called the Open SSF scorecard. Data open SSF is the open source security foundation. This is a community initiative to really level up development teams and open source. And so it tracks whether they're following various development best practices, things like code review, branch protection, you know, does it require approval to merge code into into the mainline branch, things like fuzz testing and testing as part of CI. So all of those practices, you know, you get, you get a scorecard, you get like there's like 15 things they're measuring, you get scores for each of those, we, we pull that all in. And what we've done is trained a model using machine learning to recognize the association between those input values. I just described those attributes, and vulnerability data. So what it's doing is it's looking across over 30,000, open source projects, and saying, let's look at the projects that have a clean vulnerability record, they haven't had any vulnerabilities discovered. Let's try and figure out you know, what practices they have in common, right, what was most important in terms of the scorecard practices and MTU? You know, what, what was common across those projects. And so what we find is, you know, practices, like code review are super important, which was an exciting finding, that was the most important practice, which really matches common wisdom. You know, it's, it's nice to have such a large scale data driven validation of, you know, what was sort of developer folklore, that code review is a super effective practice that everyone should be doing. And anyway, you know, at the end of

the day, we can extract from this model a score, right, and so that's what we do. And so that becomes the safety rating. And it tracks you know, not perfectly with vulnerability status, and we get 86% accuracy. But, you know, it's the most accurate score that we've evaluated so far. We want to make it even better, you know, so if people have suggestions, you know, send them our way. Basically, it's available up on Maven Central, which is the main Java repository for open source. Also on OSS index, which is an open source index of information that Sona type maintains, so you can go check it out. And you know, if you have feedback on it, there's a feedback form. We're just really excited to get this out there. Because we've done you know, lots of cool analysis in the past but but not been able to release it in a way that's I think, as actionable as the safety rating is.

14:31
Well, Steven, we've been recording now for about 20 minutes and I've been remiss there haven't used any confusing acronyms and this is Washington DC and you're standing at the Metro where you got to use confusing acronyms for something WM a TA or something. And so when I want to use is s vom software bill of ladings SPO. Isn't what it is cellphone bill of lading SBML. Is that what it is? So is this just a commercial use is using a government ID Well, what is software bill of lading or bill? I don't know. What are they? I

14:57
wish it were a bill of lading because the VNA Is connection to physical supply chains. But um, yeah, software bill of materials. Yeah. And it's, you know, what it is, is a record of all the software that goes into a particular application or library. And so, you know, I was talking about that dependency tree, you know, you have your libraries that you're pulling in, they have their dependencies and dependencies of dependencies and everything. So it's sort of that, that full picture of like, here's all the software that flows into producing this, this application at the end of the day. That's a software bill of materials. And, you know, people are really excited about these really interested in these, there's, you know, various regulations, you know, going in the works to require these, because they really increased transparency of software, you know, with open source, you can kind of reconstruct this yourself, but it's better if, you know, if it's just provided, but with proprietary software, you have no insight, you know, unless the vendor provides an S bomb, you don't know what's in that application.

And so, when log for J happened, you know, this huge sort of zero day vulnerability and commonly used Java library that affected just just a ton of applications. You know, one of the things that happened was everyone starts calling their vendors, you know, hey, Adobe, you know, do you do you have log for J, and any of the applications, we're using Amazon, you know, which of your 1500 services and API's are affected by log for j, right? Software bills of materials, provide the transparency that you would need to answer that question yourself, in theory, you know, the tools aren't necessarily there. And the processes aren't necessarily in place where you can just, you know, have a repository for all your S bombs for all your vendor applications, and just hit a button and do a scan and get the results. But I think that's the world that that we want to move to, you know, that would really make these remediation tasks much, much easier.

16:46

And this is becoming more and more prevalent discussions with people in the DoD as well. I mean, there's a guy named Nick Chalon, who reviewed and he was instrumental in coming up some very creative names for software factories, and others an army Software Factory, apparently, it's caught on it's a trending phrase, what they try to do is a sample code of Guess what? What code of the assembling you have to work with Sona type, or someone to make sure that it's in fact valid, don't they?

17:11
Yeah, that's right. You want to be scanning, you know, you want to be scanning things. And the if you think about like a physical factory, right, you want inspections, you want safety checks, you want quality checks, you know, along the line, right, all through the production line, you're on the same sort of thing with software. And so you want to be scanning software, as it's written, you want to be scanning the code that developers are writing, you want to be scanning the code that is being pulled in from the open source community. You want traceability, right, so that if, you know, if some problem is discovered, in some component, you can go figure out, you know, which applications are affected by that issue. And, and so that you can know that you're getting, you know, valid components from upstream, you know, that I mentioned, these new style supply chain attacks where, you know, these new style software supply chain attacks were unfamiliar, he can read type. Sorry, my alarm went off, and I should have

18:08
just keep going, we'll correct it.

18:12
I mentioned these new styles, software supply chain attacks were sort of fake versions of widely used projects, or uploaded by attackers. And those sorts of things can be avoided by you know, things like signed releases, validating signatures, making sure that you're getting your software from authoritative sources. And if you have that traceability throughout the software production process, then you can be sure that you know, what you get at the end of the day isn't subject to those sorts of supply chain attacks.

18:37
I have my own little company and I annotated the change my about page, I added some fun facts and insights, some fun facts for this interview. And I you know, sometimes these fun facts, quest, here's a fun fact that maybe you can tell me if it's really true or not, you know, 1.2 billion vulnerable dependencies are downloaded each month. Point that seems like a bolo, doesn't it?

18:59
Yeah, yeah, it does. And it's, it's even worse than that. Because those, those 1.2 billion, that's the number that are downloaded, that are vulnerable, but there's a fixed version available. So you don't have to be using that version. You know, there's there's some projects every now and then there's a project that has some vulnerability discovered in it. It's not patched yet. So there's just you know, there's no way to use that securely for the moment, right. Hopefully, the dev team is working on it, and you will have a patch soon. But that actually happens quite rarely. And what usually happens is there's a fix available, you're just using an old version, maybe you don't realize that this version is vulnerable. You don't have the tooling in place. You don't have that

visibility. And that's a super widespread problem. Yeah, so 1.2 vulnerable downloads that don't need to be vulnerable.

19:49
So if people are listening to this and there's snow on the ground or something, they can go to sonatype.com and get that report. What's the name of that software report? Again?

19:58
It's the state of the software supply chain rule. Well, good,

20:00
good, good, Nick, and just all throughout the year, just go there and download. Is that right?

20:04
Yeah, that's right. No, no wall, you don't have to enter your information or sign up for a mailing list or anything, just go see the report. That's perfect.

20:10
That's perfect. So when you attend conferences, what I mean, I'm trying to think would you attend cybersecurity conferences? Would it be software development conferences? Would it be dev SEC ops? I mean, it's hard to classify you. So you may not know what to what's the what category?

20:25
Yeah, I'm hit up all of them really, you know, and you see, these are all coming together to with, you mentioned dev sec. Ops, right. So, you know, developers and security, you know, their concerns and work is sort of merging, in essence. And so yeah, we certainly talk a lot to the security community, but also work a lot with developers with the DevOps community, because that's really where this all comes together with the automation piece to make sure that the scanning is happening in a, you know, predictable and automated way. You know, I think everyone needs to be aware of this. And there's different parts of the problem that land in different people's courts, you know, developers are more responsible for keeping things up to date, if there's a code change that needed to make to adapt to a vulnerability that's on their plate security team maybe is more involved in evaluating new software that they want to pull in new dependencies, new libraries, they might like to use, you know, architecture concerns, you know, are we using this in a way that that is protected, that, you know, we have sort of controls in place where we need them, everyone has a role to play.

21:27
I teach a little school downtown, I sit down with six software developer types, and, and what's what's interesting is that they're in demand, and they know it, and they're interested in job satisfaction. And it's a topic that didn't come up in my generation, but they're literally and I would think, just from talking to my students and talking to people in this business, that if they had a better software Supply Chain system, it would make it would make them more satisfied their job and and wants to stay and and be more productive and not bolt and have to find

someone else. I mean, this is a challenge all federal agencies have keeping people happy. And soda companies like Sona type, you find someone good, and how do you keep them down on the farm? Because they can get jobs like an instance? So this, this, this cybersecurity software development in HR? Steven, are you in HR? Now?

22:17
I haven't attended any HR conferences yet. Yeah. But we did do a survey this year as part of the report 662 software development professionals at varying companies of various sizes, asking them a number of questions about their software development practices, the outcomes, they see initiatives they have in place at the corporate level. And a couple of interesting things that we found relating to job satisfaction, we're, that number one, more mature software supply chain practices are associated with higher levels of job satisfaction. So you know, it makes sense, right, developers want to work at a place that's working well, you know, where they can, they can focus on their job, they can focus on writing code, they can focus on what they love, and, you know, the tools and processes are in place to manage all of this other stuff that, you know, they just at the end of the day, don't want to get blamed for, right, nobody wants to be responsible for, you know, data leak or something. The other thing, interestingly, that tracks with job satisfaction is engagement with open source. So you know, a lot of companies use open source, far fewer companies contribute back to open source. But that sort of support corporate support for upstreaming, which is the process of pushing your your modifications back to an open source project. That is also strongly associated with job satisfaction.

23:38
If you're listening to this interview and want to approach software development from a different perspective, you may want to listen, Episode 13 is called avoiding cloud collisions. And it talks about managing stuff in the cloud, we had to complete different opposites here, we had one and the other, we go back and forth here. And but it's a fun interview with a company called Data dog. Great. So Steven, if, if we're waiting for the Metro in Alexandria, and I happen to chat with you, and what do you do? Oh, some type? Oh, great. So so my question is, you always have other companies that compete with you. So what? So I guess the question is, I mean, what did people get wrong about managing the software? Is that go far enough deep in the tree? Or what problems that What mistakes did they make and try to manage this open source software?

24:22
Yeah, yeah, there's definitely sort of better and worse, or at least, you know, more thorough and less thorough ways of, of looking into into your supply chain and finding issues there. So you mentioned depth, you know, that there's certainly variations between tools and in terms of how deeply they look at the dependency graph, or how they construct the dependency graph, I should say. So, you know, common approach is just to look at the source code in the repository. Look at the, you know, the build manifest and see, you know, here's here's the declared dependencies is what this application says it uses right and what's Important is to recognize that that's not the whole story, right. So when you package things up, you know, you're done building it, your package it, you maybe put it in a container, and then you push it out to development, a lot of software gets pulled in along the way. And so you want something that's doing scan, we call binary scanning, at least for Java, you want something that's scanning the bytecode, that's generated, not that not the source code level, because what we see is a vulnerable component, like the log for j component, for example, it doesn't just

appear in the log for J project, but actually appears in hundreds of other projects, you'll find that same component with that same issue. And so by code scanning can help identify that, the other big thing to look at is the vulnerability database. So you know, you can, step one is construct the bill of materials, see what you're using step two is see which of those are vulnerable, that involves vulnerability data, which you know, is publicly available, but it's not the whole story, or it's not as precise as it could be. So there's the National Vulnerability Database, which, again, is what a lot of tools pull from, they just use that. But what you'll find is that often, you know, that's sort of imprecise about exactly what versions are affected, you know, it says everything before version 3.2. And but actually, it's only between version 2.5 and 3.2, because the vulnerable code wasn't even introduced until version 2.5. So before that, you're, you're safe. And and so you know, what we do, we have a large data security team security data team that goes and does deep dives, you know, whenever there's one of these advisories, they go, and they look at the code, and they say, you know, here's the commit where this code was introduced, here's the commit that fixed it, here's the versions associated releases associated with those, and make sure that we have really accurate data about what's affected by this vulnerability. And that that's important for cutting down you were talking about, you know, there's all this security to keep track of, you know, there's so many updates, so many things requiring attention, you know, you scan an application, you get a huge list of things to deal with, the more accurate the data, the shorter that list is right, that helps you avoid these, what are called false positives, where, you know, it's saying this is vulnerable, when it's actually not, you know, you want to make sure you're not spending your time chasing those down. Because if it doesn't affect you, you know, you shouldn't be spending development time on it.

27:15
I come from a liberal arts background. And a few years back, I was talking to a bunch of software developers, and they love to argue, and they're talking about artifact and I said, Oh, something I know about your cultural artifacts. Now, John has nothing to do with anthropology. Yeah, artifacts. Oh, sorry. But it sounds like anthropology, but isn't, but it may have a bearing on this discussion. So you talked about binary libraries, artifacts part of this, or is this just kind of superficial thing?

27:41
Yeah, I mean, artifacts, like, you'll often hear the term artifact used for, you know, those binaries, for, you know, containers that you're creating sort of anything that it's, it's gone through their production process, right, it's ready to be deployed. And so a lot of companies will use artifact servers, artifact caches, which are a place where you can you can do that build, you can put that artifact, it's like putting it on the shelf, you know, it's ready to be used. And then it's much easier and faster to grab it, and use it in further, you know, software development practices. And so that's actually a really great place to put controls in. And so we have, we have an artifact server called Nexus repository. And we have in within we have software that goes with that. And also Artifactory is the other common one. And you know, and so we have product called firewall that interacts with those artifacts servers, and will actually block at at the boundary, vulnerable dependencies, I get this question a lot. So Sona type manages Maven Central, which is the main Java repository, you know, it's where people go for most Java open source projects. And when log for J happened, people were asking me on Twitter, you know, why? Why doesn't Sona type, just block the vulnerable versions, right, don't let anyone download a vulnerable version of logger J? Well, that would break a lot of bills that would cause a lot of chaos, it would sort of, you know, be antithetical to some of the, you know, promises that we make as maintainers of

that repository, that things will always be available. And you know, we won't interfere with your build process, and so forth. But, but that's something you can totally do at the corporate level, or at the, you know, government organizational level, you can say, you know, we're just going to have a policy where, you know, if it's a severe enough vulnerability, we won't let it flow into the development process. And so that artifact server, you know, that's a place where you can put that into place with products like firewall, it also helps you block these new style supply chain attacks I mentioned. So you know, malicious code injection, right? That complicates the update calculus a lot because it used to be, I just need to stay up to date. I just need to be using the latest version all the time and I'll be as safe as I can be. But now that latest version might be tampered with, you know, it might have had an attack against it. That was is injected intentionally. And so we have something called release integrity, which uses machine learning to monitor for anomalies in in commits and say, hey, you know, this new version, it looks kind of suspicious. So maybe we should wait a few days before we let developers use it and see, you know, if anything's discovered, or if it, you know, passes some sort of vetting process. Well, I

30:21
didn't realize that artifacts were that big of a part of the story. You know, I thought, I've heard people argue about them enough. And they were like, they were like, oh, like, get a nine sixteenths wrench and then use it another, why should they reinvent it? And they can use it again. But it has a built in vulnerability and reuse it, then it just adds to the problem, doesn't it?

30:38
Yeah, yeah. That's right. That's right. You want to you want good visibility into what's going into those artifacts, servers, because the whole point is, things are there so they can be easily reused. So that's, that's going to get you know, it's going to fan out into multiple parts of your organization.

30:54
So are you speaking at conferences the next year or so or anything coming up that our listeners should attend to learn more about software libraries and controlling all this information?

31:03
Yeah, I am. I'm speaking at all day DevOps on? It's November 10.

31:12
Yeah, and actually, let me let me get the actual date. And I'll answer that question again.

31:18
He's actually speaking all day long. He's not gonna go for eight hours. There's just all day all day long. You have different topics are brought up. I think Derek weeks is involved in that a lot of people are fact a lot of people started to get started. Yeah, see? Well, they can go to all day DevOps and find out, get more information, but that's good to know. So prediction.

31:38
Can I shoot? Let me just answer that again. Sure. Yeah. Sorry. Yeah. So sorry. Yeah. So I'm, I'm speaking at all day DevOps, which is coming up next month, I also just gave a talk at the DevOps enterprise Summit, which had, there's a recording of that, and you can you can sign up for that video library and see that talk. Both of those are on this software supply chain report. And so I'll be going into great detail on sort of all the findings as well as the best practices that come out of that that suggestions.

32:15
Final questions? Always a crystal ball question. I think there's going to be an incident that's going to happen and cause a lot a shake up. And this was five years from now you think he's going to slowly make progress or is going to be flat and then an incident and make rapid progress? Like, like remote for COVID? So what do you think is going to happen with people adopting more careful principles with reusing software?

32:36
Yeah, I think we'll continue to see more and more attacks here. I mean, in part because cybersecurity attacks are becoming so lucrative, you know, with things like ransomware Bitcoin mining to these give attacker is a direct way to monetize, you know, their activities, right. And so it really ups the incentives for people to go find problems in software, you know, at the same time, I think that, you know, the defense is getting better. But, but, you know, it's not going away anytime soon. You know, things like log for J, these super high profile, you know, high impact issues. They're sort of unpredictable. You know, there's some randomness involved there. So, you know, is it a, you know, three years from now, is it five years from now is that next year? Hard to say, but I'm sure there will be another one.

33:26
Yeah, it's a safe prediction. I'd say. Well, we're running out of time, unfortunately, here. You've been listening to the federal tech podcast with John Gilroy. I'd like to thank my guest, Dr. Steven McGill, Vice President product innovation at Sonatype s o n a t ype. Thanks, Steven.

33:42
Thank you.