EPISODE 971

[INTRODUCTION]

[00:00:00] JM: The software supply chain includes cloud infrastructure, on-prem proprietary solutions, APIs, programming languages, networking products and open source software. Each of these software categories has its own security vulnerabilities, and each category has tools that can help protect a company from attackers that are trying to exploit those known vulnerabilities.

As open source software has grown in popularity it has turned into an enormous potential attack surface that is difficult to protect. Snyk, S-N-Y-K, if you have not known how to pronounce that word, is a company that builds security tools for companies that are consuming open source.

Guy Podjarny is the CEO of sneak and he joins the show to discuss the security vulnerabilities of open source projects and how his business works. Guy was previously the CTO of Akamai. So he has significant experience in technical leadership. He's also the host of the podcast The Secure Developer, which I recommend for anyone who's interested in technical interviews about security topics. This was a great conversation. I definitely want to have Guy back on the show in the near future because there was so much that we didn't explore. So we'll definitely do that in the near future.

Also, I want to mention, we are hiring for two roles. We're hiring a content writer. If you're interested in writing about software engineering and computer science, perhaps you have an academic bent or a journalistic bent. You can send me an email, jeff@softwareengineeringdaily.com. We are also hiring an operations lead. If you want to figure out how a business works and you want to help us improve our business, both of these roles will work closely with myself and Erica. If you're interested in them, you can just send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:02:03] JM: If your product has dashboards and reports, you know the importance of making those analytics products beautiful. Logi Analytics gives you embedded analytics and rich visualizations. You don't need to be a designer to get great analytics in your product.

According to the Gartner analyst firm, the look and feel of embedded analytics has a direct impact on how end-users perceive your application. Go to logianalytics.com/sedaily to access 17 easy changes that will transform your dashboards. That L-O-G-Ianalytics.com/sedaily. Logi Analytics is a leading development platform for embedded dashboards and reports, and Logi gives you complete control to create your own analytics experience.

Logi Analytics has been a sponsor of Software Engineering Daily for a while, and we're very happy to have them. So thanks to Logi Analytics, and go to L-O-G-Ianalytics.com/sedaily to find 17 easy changes that will transform your dashboards. You can get better dashboards and reports inside your product with embedded analytics from Logi Analytics.

[INTERVIEW]

[00:03:35] JM: Guy Podjarny, welcome to Software Engineering Daily.

[00:03:37] GP: Thanks for having me.

[00:03:38] JM: If I am a late stage startup or an enterprise, I'm getting my software from so many places. I'm getting it from on-prem, cloud, closed source, open source. Describe the modern software supply chain.

[00:03:55] GP: Ooh! That's sort of a big question. I think, in general today, it depends on who is the consumer of that software, right? So you've got sort of maybe the central sort of SysAdmin sort of IT ops. Maybe a little bit more old-school, but still very much reality today that might run sort of enterprise software, right?

So it's not about the applications you build necessarily, but things that the organization needs to survive, right? So pieces of software is still to me primarily – Still primarily purchased. Whether it's probably the big trend there is sort of software as a service and all that might. You might be

running on-premise and you'd run on those, but they're still primarily vendored. I don't think open source quite dominated that area quite yet, maybe. Maybe databases are kind of exception to that statement.

Then there's a software you build, sort of the application that you build. It doesn't really matter actually if you're sort of a startup or a large organization, your own developers as they build it. I think over there really is where a lot of the changes happen. Over there, when you think about modern development today, most of the software, the components you do are really open source. It seems quite odd today to like buy an SDK or something like that. You'd be buying. You would have vendors that are platform vendors about worries that you run.

So you might get some managed Kubernetes from something or the likes and you might still sort of license software as a service for sort of surrounding services and you do that a lot, whether it's the cloud platform again or whatever it is, Auth0 for authentication or some host the database or things like that, and tooling.

So that's very much very common, these third-party services that you need to run around. In your software itself, it's pretty much all open source. So I think when you look at kind of that constellation, the key difference between that delineating line – I mean, yes, there's like technical. Do you use a service? Do you use software to do – But the key difference is actually did you buy it or did you just downloaded it off the Internet? If you do, if you use open source, which is amazing and spectacular and rising, the shift or the real changes that people need to own it.

When you download a library from the Internet or some piece of software, whether it's to connect your app to Facebook, or it's a database, or anything like that, there's nobody there to sort of have your back. The community is there. They're going to educate you. They're going to kind of help you maybe use it correctly. But fundamentally, you own it. You downloaded it. You own it.

So I think when you talk about kind of maybe software management and sort of understanding what do you need to do for these – These open source components are this weird sort of a hybrid, right? The third-party software. It's not software that you've written. You don't know it

very well. You're not going to edit it. But there's no vendor around it. So I would say kind of in – I deal a lot with sort of this notion of the software supply chain and sort of open source security. In that space, the realization that I think is sinking better and better is that you need to invest within your development process, within your tooling, within your company in having the right kind of foundation to deal with owning those bits of software.

[00:06:59] JM: Open source has been a growing volume of the software that has been consumed. What are the vulnerabilities in this open-source section of the supply chain?

[00:07:15] GP: Vulnerabilities in open source are just like your everyday vulnerability, right? There is nothing really neither more secure nor less secure in open source components. They can range from some cross-site scripting, vulnerability, frontend component to an SQL injection to something that's behind to a remote command execution. All the same vulnerabilities you can write in your code.

The primary difference is in prevalence. So first of all, when you look at an application today, you look at the amount of codes deployed. 99% of the code deployed is not code you've written, right? You've written an app. You've used 10 libraries. They pulled in 500 others in a cascading order. Generally speaking, you're getting a ton of value that way very, very quickly. But you're also getting a ton of codes, and that code has bugs, and some of those bugs are vulnerabilities and are security bugs.

So they range from severe to benign, sometimes exploitable, sometimes not exploitable. But the vast majority of your code, and with that, the vast majority of your risk, comes from those components. SO that's sort of one way to look at it, which is these vulnerabilities, they vary, but they primarily come from the open source components you use.

The other angle to that is the attacker's perspective. So if I want to break into your app, I can do a lot of heavy lifting and figure out the bits of code that you've written and find vulnerabilities in it. If I've succeeded and have evaded detection in the process, then I can hack your site or sort of your application.

While if I found a vulnerability and open SSL, off the bat, right away, a third of the web or a quarter of the web is suddenly vulnerable to these components. So these components are more easily found. I can just sort of find that library. I can disassemble it. I can kind of reverse engineer it. Find a vulnerability in it without anybody, without any risk of capture. Then once I found a vulnerability, I can actually exploit it in many, many, many very different victims.

The last step for that processes is I oftentimes don't even need to find that vulnerability, because security hygiene at scale is hard. So if I am not actually targeting a specific organization, but I'm just looking for victims that I can steal stuff from, or whatever. I can run some Bitcoin mining on. Then what I may do is just look at the known vulnerabilities that have already been discovered, especially ones that have been discovered recently. Run an exploit for them.

Once again, I probably don't even need to do that. I can just find one on the Internet or buy it on the dark web. Then I can just run this canvas exploits, and I will find a lot of vulnerabilities. So it's really about that prevalence of open source code, both its percentage in your app and it sort of prevalence across customers that makes open source vulnerabilities more dangerous. Not that they themselves sort of open source code is neither more or less secure. Does that make sense?

[00:10:01] JM: Are you saying that because some of these open source libraries are so popular, it makes it more valuable for people to find vulnerabilities within them and potentially sell those vulnerabilities, or just use them as essentially zero days.

[00:10:18] GP: Basically to just exploit them. Yes, it is more valuable for attackers to sort of find them. But I'm saying even beyond that, most attacks that attack open source components, the ones performing the attack are not the ones that found the vulnerability. These vulnerabilities, it's all very public, right? So you see that we can – I don't know if we can mention Shellshock. Shellshock was this like very big vulnerability in Bash that came out – I don't know. I want to say about three or four years, maybe five years ago. Super severe remote command execution vulnerability in Bash.

If you had quite a common set of constellations, then you can just break into servers quite easily. It's a severe vulnerability, but it's nothing really terribly unique. What's unique is that Bash

is everywhere. So within two days, you saw botnets deploying exploits of Shellshock. Shellshock was discovered. I forget who it is that sort of found the exact that disclose it. But it was legitimately found by researchers who disclose it responsibly, and if you updated to the latest version of Bash, you would be secure. But updating is hard, and updating at scale is hard.

So really it's this race, and what attackers find compelling is, yes, they can find vulnerabilities and use them in many ways. But oftentimes they don't even need to find them. They just need to either write an export or use an exploit, but it's all about doing that quickly. Then they'll find a lot of victims to be able to exploit. So it's just attackers are lazy, just like all of us, and it's easier for them to find victims that way.

[00:11:44] JM: You're the founder of Snyk. Explain what Snyk does.

[00:11:48] GP: Snyk is a dev-first security company or sort of solution. Really, our premise is as of software development changes, and more and more responsibility and capabilities get built into the app. A lot of the security responsibility of those shifts into the developers. So whether you think about the operating system that's now in containers and previously was in VMs, right? If you think about network configuration, that might be like a VPC or some data center configuration and now might be microservices configuration you define in your Helm charts.

Of course, again, open source library as an app code. All of those things combined for a broader app, and that app is managed by developers and they cannot be slowed down. That's the business demand. So our belief is that the only way to scale security in this brave new world, into an extent, the old one, is to get developers to embrace security.

So we built a company that is a developer tooling company that tackle security, and we started with core solution around securing open source. So helping you figure out which components, which open source libraries to use. Are they – First of all, what they're using? Are they vulnerable? Then most importantly, help you fix that without slowing down development, and that product has gotten kind of great developer adaption. Has some 500,000 developers using the product. Sort of bottom up free for open source projects to use. So kind of the product keeps us honest, right? If it's not good, people won't use it. So really building a developer-minded product.

Then we've expanded. We're about four-years-old. So about a year and a bit ago, we started veering into also expanding into container security, and we actually have just now launched a standalone container security. Once again, having that sort of dev-first security mindset to it. Thinking of the container as this evolution of the app. Not so much the evolution of the VM and thinking about how can you kind of wrangle these Docker files and how do you sort of patch containers as a part of your build process and do all of those things in a way that is kind of a first-class citizen and is pleasant for developer to use.

A lot of people sign the check are security or some platform teams that are a bit governance-minded, but even they, the most important thing for them is that developers embrace it, because if they don't, it becomes shelfware.

[00:14:00] JM: That's bottoms up entry point developer tool that 500,000 are using, what is that? What does it do?

[00:14:06] GP: It's about securing open source. So while the whole thing has a freemium component to it. So you connect it to your git repositories. It can connect in many places, but we're sort of best known for the git connectors. So you'd say, "Hey, connect to my, say, GitHub repositories." It will kind of crawl the repos that you've opted. It will find package JSON files, POM XML files, GEM files, whatever it is that the dependency language that you use, and we'll expand the graph of dependencies that those file will generate. Then we'll will intersect that with a vulnerability database.

We maintain a vulnerability database. It's kind of a topic on its own right. If we want, we can dig in. But we basically keep a database of known vulnerabilities in the ecosystem. Then we simply put, intersect the dependencies we found that you're using and the vulnerabilities we know exist and we'll tell you which ones are vulnerable.

Probably our biggest claim to fame is once you found a vulnerability and we told you, "Hey, you're using library A that uses library B, that uses library C. C is vulnerable." We can tell you, "Well, if you just upgrade A to this minimum upgrade, it would trickle down to this upgrade to B, and to this upgrade to C, and you'll be safe. That's the way to fix it." We packaged it up as a fix

pull request, and we opened that up to your repository alongside of course any kind of email notifications and the likes.

So this to me is one example, maybe the strongest one of developer mindness. Developers jobs isn't to find issues. It's to fix issues. So if you want a developer to use a security solution like this, "If you just make work for me, I'm not going to like you too much." What I want you to do as a solution is to take load off me, off my plate. So this remediation, this automated remediation was a core component of it.

Yeah. So that's the rambling a little bit. The core product is really around that. It's figure out. Connects to my repos. We have a CLI, gets a million a month. Very widely used. That's more the Swiss Army knife to get integrations a bit more opinionated. Then helping find out which libraries are used where and then help me fix vulnerabilities in them in an ongoing fashion. When you get into the premium land, you get into governance components and understanding what's used where in mass and guiding policies and license compliance and the likes.

[SPONSOR MESSAGE]

[00:16:20] JM: If you are a SaaS or software vendor looking to modernize your application distribution to gain more enterprise adoption, checkout replicated.com. Replicated provides tools to deliver your Kubernetes-based application to enterprise customers as a modern on-prem private instance. That means your customers will be able to install and update your application just about anywhere.

Bare metal servers in a cloud VPC, GovCloud and their own Kubernetes cluster, vSphere. This is a secure way the your customers can use your application without ever having to send data outside of their control. Instead of your customer sending their data to you, you send your application to your customer.

Now, this might sound difficult and maybe you're not used to it because you're a SaaS vendor. You're a software vendor, but Replicated promises that recent advancements from tools like Kubernetes make it far easier than before, and the Replicated tools can help vendors operationalize and scale this process.

The Replicated tools are already trusted by noteworthy customers like HashiCorp, CircleCI, Sneak and many others. As a result, over 45 of the Fortune 100 already have an application deployed via Replicated in their infrastructure. That's a strong sign of adaption.

Go to replicated.com for a 30-day trial of the full Replicated platform. You can also listen to an interview with Grant Miller, the CEO of Replicated, that we did a while ago.

Thank you to Replicated for being a sponsor of Software Engineering Daily, and you can check it out for yourself at replicated.com and get a free 30-day trial.

[INTERVIEW CONTINUED]

[00:18:28] JM: The auto-generated update to my git repo that suggests the package that I should update to that will clear away my security vulnerability, how universally does that work? Because I mean sometimes if you update from – Like this is why people get stuck on like Java 6 and like, "I want to update to Java 7 across the organization, but I'm really scared, because maybe it's deprecating something. Maybe it's going to add some latency. Maybe the garbage collector works differently. I don't want to touch it until I am absolutely ready."

So how readily are people just merging in those suggested pull requests?

[00:19:07] GP: Yeah, I think it – So it depends. It's a really good question, and fundamentally this is all one being exercise in risk management, right? If you don't upgrade, there is a risk. If there's upgrade, there is a risk. The first thing we're trying to help you do is understand that risk tradeoff, right? If Shellshock is sort of present, right? If you don't upgrade, then you're going to get owned by a botnet tomorrow. Then sometimes even taking the system offline is better, is a lower risk than doing it, let alone, an upgrade.

So what we first tried to do is we try to inform you. Then the second thing is we try to introduce the understanding of how complex it is next to it. Most organizations work as they do an audit. Whether it's a thing that's built into your pipeline or it's a side thing, and it will these gazillion

vulnerabilities. Then only the very top ones will get triaged, which is this painful process where some security expertise and application expertise need to combine to decide if this is worthy of fixing. All that process happens and gets into increasingly small number of vulnerabilities just because of how expensive and slow it is. Then a subset of them would be logged as some Jira ticket. Only then somebody would figure out what it takes to make that vulnerability go away.

So what we tried to do is we tried to make it easier to fix than to triage in many cases. So when you look at customers that embrace it, how easily or not varies based on ease of upgrade and severity of the issue. If it's a severe vulnerability that has a patch upgrade, people embrace it very quickly. If it's a major upgrade, it very clearly slows things down. Now there's a conversation. If there is a very severe or a critical sort of a CVSS 10 – CVSS is a standard severity score. If it's a CVSS 10 vulnerability, even if it's a major upgrade, you might pull the cord and sort of stop the factory line to fix the issue in place.

So we see other variety of like another variant might be. How good are your automated tests in this systems. Indeed, some systems are more fragile than others. How sometimes it's just literally the bias for updates that one development team might have versus another.

[00:21:17] JM: So this kind of scanning of my git repos, this could be useful as a cron job. It could be useful as something in my CICD pipeline. How does that scan fit into a developer's workflow?

[00:21:35] GP: Yeah. I think the full answer is all of the above, but it boils down to different values at different stages of development. So I kind of take two stances of it. One is developers are these beautiful people. I'm a developer. I'm kind of allowed to comment that way, which is on one side, you want tools to just work. You come in, we've got like 30 seconds ADHD to make an impression and it needs to just work and I don't care how complicated my environment is.

On the other side, once it works, it needs to be fully customizable and fully adaptable. So the way we tackle those two is we have opinionated integrations into things like git and connected it to Kubernetes, and they work in a way that just works, next, next, next. Just kind of works. We have this commandline interface and API that are the Swiss Army knives. So you can adapt it to your surroundings.

Those two paths allow us to adapt to the always reasonably unique different development processes that people employ. Then with those tools, we tend to connect in three places. The source code is closest to the fix. So whether it's git or an IDE plugin, which we have a bunch of as well, those are the earliest places where you can find an issue. They're most interactive. So you're writing code or you chose to whatever you open the pull request. It will tell you, "Hey, you've introduced the vulnerability. It's the great places," and you can open a fixed BR, right? So that's closest to the fix and earliest detection. So lowest cost.

The pipeline, mostly the CICD sort of systems, are really where you put policy enforcement. It's this sort of thou shall not pass type elements, but breaking a build is a pretty hefty action. So again, if it's a supercritical vulnerability – Or we also do license audits. So if it's like a GPL component, you might say, "I don't care. Break the build." It's okay that everybody stops to fix this thing.

Then last but not least is we monitor production systems. So we'll connect to lambda to see which functions you have there, or to Kubernetes to see which containers you run, and a variety of others. That's kind of closest to the fact, because sometimes, especially in container land, you have blue-green trains. You have multiple versions running on it, or you might have – You pull down some nginx image from Docker hub. So it may never have gone through your pipeline.

So kind of the monitoring, what's actually deployed in production is closest to the fact, and it's, "Well, there's a vulnerable kind of library here, a vulnerable container. I don't really care how it got there. You need to deal with it." It's also a place for us to get insights and now say, "Okay. Now we know that this container is deployed in this publicly accessible critical system. We use that information to help you prioritize the vulnerabilities we found early on in the git," to say, "Well, this vulnerability, you really need to tackle right away."

We even do some runtime insights. So you can, for a few of the languages, like Java, JavaScript, what you can do is you can actually use Synk in runtime to figure out which libraries or which pieces of code from every library you're actually executing in runtime. Once again, it helps you prioritize to say, "Hey, we found this vulnerability in the source code, or in your

dependencies, during development time. We know that this vulnerability, you're actually invoking the vulnerable code in production, which is not necessarily the case for all the ones we find. So you should deal with that one more urgently than the others."

[00:24:58] JM: The strategic advantage of Snyk, or the moat, you might say, or at least one of the moats, is your vulnerability database. So there are these public vulnerability databases where people can go and find out at any time, "Here are some known vulnerabilities across the Internet."

What's the difference between your vulnerability database and the public vulnerability databases?

[00:25:26] GP: So, again, kind of a good question here. So just establishing a little bit of knowledge base, right? So vulnerability is the most standard way to find out about vulnerabilities, is the NVD database, the natural vulnerability database, with CVEs assigned to different vulnerabilities, common vulnerability enumeration, I think, or vulnerabilities and exposures. Sorry.

That system was designed really for, whatever, Cisco appliance that has a vulnerability and they need to have some standard way of getting an ID there and getting it to all their customers. It's a slightly laborsome process to – First, you need to know how to – Like that you need to get a CVE assigned. How to do it? You need to debate a little bit with different CNAs, CVE numbering authorities on is this a real vulnerability? Is it really a topic? What the severity of it?

Sometimes if I find a vulnerability in your library, there's a little bit of an ownership. Am I allowed to find a vulnerability in your library and just disclose it? Sometimes you need to like prove ownership, "It's my library. I'm allowed to report a vulnerability on it." It's a complicated process.

It's definitely not designed for open source library maintainers. They don't know any of the above. They have no incentive to really go often and go through this process even if they know it exists. So the reality is when fixes happen in open source land, maintainer would fix the vulnerability and stay as much in the release notes, and that kind of disappears into the ether of GitHub.

So what we do is we have an internal system, which we kind of think of as a threat into a system that listens to open source activity. So it listens to all sorts of GitHub activities. It listens to a [inaudible 00:27:01] or a project, to the Red Hat forums, to all sort of research or social feeds. We glean, handle the vulnerabilities, and we funnel them to analysts, to real people, that look at them and say, “Is this a real vulnerability or not?” If it is, then they figure out what it is. Okay. So what’s the vulnerability? What's the severity? Here’s metadata explaining it. Here are the libraries and the versions that it's relating to, and they put that into the database.

We also do – We have other sources to the database, like we do a cleanup exercise for CVEs. The NVD is messy. It has a lot of these vulnerabilities. Many of them have very incomplete, like very unstructured type of data. So we cleaned it up. We have better data on it. Also, we do our own research. We collaborate with a whole set of universities right now on various research projects. So all of those end up uncovering more vulnerabilities.

When we find a vulnerability, we go through a responsible disclosure process with the entities in place. We also do that as a service to the community. So if you find a vulnerability in a library and you kind of don't know what to do with it now. You can come to us and we get several hundreds of those every quarter where somebody would reach out to us and will take the burden of doing the responsible disclosure.

So all of those amount to this clean, high-quality database of known vulnerabilities that is timely. Sort of how vulnerabilities oftentimes before there’s ever a CVE, if it ever exists. It's more comprehensive. It’s cleaner quality. The database is primarily kind of a core component of the product, but then some kind of cloud giants, some big financial institutions. Recently, like some players, like recently Trend Micro, which is a security player. There are a few others. Actually, license just the Intel. We call it Snyk Intel. It’s like just sort of that vulnerability database, almost its own product.

[00:28:51] JM: Okay. How many people does it take to maintain that database and how much infrastructure we had to develop around the practices of updating the database and prioritizing which vulnerabilities to research more?

[00:29:07] GP: Yeah. It's an ongoing exercise. So we have, I think, today probably about a 20-person security group, and they are not quite half-and-half. I'd say maybe about close to half-and-half, engineering and analysts. So the engineering side, and then some researchers.

The engineering side is – Really, what they do is they build a system then increasingly, like everything, we want to scale with tech. That increasingly listens to more and more sources of information, and also improves the analysis as much as possible to basically hand the analysts on a silver platter. Here's all the information. You just need to sort of say yes, right? Even when you compose an advisory or things like that, can you pick and choose pieces of it? Is there an expert in the wild? A lot of those sources of information.

Sometimes, when there's enough confidence, the vulnerability actually really just get a quick scan, right? A quick kind of visual inspection. So those two teams work together, where basically the engineering team is – Like this has been for four years now. We've been evolving and honing this system, right? So the system allows us to scan more and more of the Internet and find more and more vulnerabilities in looser and looser structures, but then similarly, they allow the analysts to be able to do work more efficiently. So the analyst team is definitely been growing, but not linearly to the output that they provide.

So those teams kind of work together to produce something that is a quality output, and research is generally more inspiration. Research, we can't compete with the world. We're not going to find more vulnerabilities than the world does. But what we do do is when we do a piece of research, we'll find and we'll disclose some vulnerabilities, but then we use that knowledge to build better rules into the system to find other patterns, other candidates that help us just sort of evolve and improve what to look for.

[00:30:58] JM: When you're talking about looking for things, it sounds like it's a combination of automated and manual processes. Can you do scan Stack overflow posts and you like look for people saying, "Look. I don't really want to use this Apache Struts thing anymore. I'm sensing some vulnerabilities in it," or you look through GitHub comments of some new JavaScript library and can do some natural language processing and say, "Oh! These people are talking about some potential cross-side scripting in this new JavaScript library."

Maybe we should assign an analyst to look into this further, because we've seen – From NLP, we've seen some hints that should this contain a vulnerability? Because the boots on the ground developers are talking about this potential vulnerability. So I'm just bringing up. I have no idea if this is what you do. Am I in the right direction? Is this the kind of stuff that you can do to sort of hunt down the potential vulnerabilities across the Internet?

[00:31:56] GP: Yeah. Spot on. That's some of many kind of approaches that we'll take. It might be as simple as somebody opened a GitHub issue or a GitHub poll request with the fix, and the title of that is SQL injection and da-da-da-da-da. So that's like a very, very simple version of it. But still, if you didn't weed it out, you wouldn't do it, and it goes all the way to indeed saying –

[00:32:17] JM: Because it hasn't emerged yet.

[00:32:18] GP: Yeah. Here's like a forum conversation and they're talking about, "Hey, when I run this server, it keeps crashing and it consistently – When I do this, it keeps crashing." They might have not even realized it's a vulnerability, and we'll funnel it down. That's when I was referring to the system getting better and better, is when we started, we're really pretty much finding the former. It was more about just uncovering bits of information that we're just not centrally located.

Then as we get better and better, then we're able to find more things that are more elusive. What we don't do really is we don't really look for zero days. We do it in the research bit, but we generally don't focus heavily on finding vulnerabilities that are not yet known. We do it as research, but I guess our view is that companies are struggling today as it stands to sort of cope with the unknown vulnerabilities. So we focus our efforts on getting that nailed, because we see it as a higher priority than finding the more of the unknown vulnerabilities, which matters as well. It's just not as urgent.

[00:33:22] JM: When you get something like that example you mentioned where somebody has opened a new GitHub issue that is a pull request fixing some SQL injection thing, but it hasn't been merged in with the CodeBase yet. So there's not actually a new version of software that – There is no fix that's been issued yet. You can scan that GitHub repo. You can find that pull request to fix a vulnerability. You can add that to your database, but there's no remediation.

There's no suggestion to give to a developer. So what do you do in that situation where you add it to your vulnerability database? Some developer that's using that package runs the scan. They find, "Oh! This thing has a vulnerability, but there's no fix issued yet." What do you tell them?

[00:34:13] GP: It's a tough situation, and that is unfortunately reasonably common. If you have a library and it has a vulnerability, generally, you want to know, and there are oftentimes other ways that you can fix it. But again, it goes all away from like this is so severe. You will take the system down, right? This is a severe vulnerability. It is now known. Your system is dealing with sort of private customer data. You would actually prefer downtime in an extreme case to having it, to oftentimes many other ways for you just to sort of sanitize the input coming into this library, or otherwise separate it from the network and soak in such capacity. Maybe it opened a port. So it depends on the vulnerabilities. They're oftentimes mitigating controls that you can put.

What we try to do is we try to inform you about them in our advisory, right? So we'll tell you, "Hey, there is this issue here. You should know about it," and here's what you can do about it. That statement, everything I just said, is absolutely true when you have one vulnerability. Now, when you have a hundred, then a very natural tendency is to start from the ones that are fixable.

So it depends on sort of the stage that you're at when you're dealing with the vulnerabilities. If you're just getting started, there's a bunch of flags in the product that you can turn on to basically only show you fixable vulnerabilities, because often times those are the ones that you just need to just get going. Do something with them.

But as you kind of get a slightly better handle of the vulnerabilities in your system and you're kind of ready to level up, if you will, then really this is information that you should know and you should make a conscious decision on how to handle.

Now off to the side, we, for JavaScript only, and this is an interesting kind of something the ecosystem is still digesting. We offer patches. A more benign version of what you've just described is I'm using library A and it uses library B, and B had a vulnerability and fixed it. All good. But there is no version of A that gets me to the safe version of B. The dependency chains take a while to catch up, and sometimes they're stale and they don't catch up.

So for those cases, what we've done in JavaScript is that we actually built patches. So we took the original fix would. We packaged it up as a patch. We kind of stripped away any other unnecessary changes. We back port it to old versions, and we allow you to use a command called Snyk Protect to patch that. From the base, what you can do, like you're in an app and you use A that uses B. You can't upgrade B, but what you can do is you can patch B. You'll install the vulnerable version of B and then you'll run Snyk Protect and it would patch that vulnerability. So literally, is a patch file. Like you go in, find the relevant file and patch it.

We're very, very diligent to ensure that we don't introduce any breaking changes or anything like that to those libraries. In fact, if there is a risk of breakage, we don't create a patch. We kind of take that as a limitation. We apply today I think about 600,000 patches a month for different vulnerabilities, and oftentimes that's the way to address it.

So for JavaScript, we do that. Then the last bit, getting a little bit here on my soapbox, but the last bit that we do is we keep monitoring the system for you. So if you used A that uses B and there is no vulnerability or no upgrade to get you to the safe version of B, we keep looking. Every day, we're going to run another test. If a new version of A came out that fixes that vulnerability, are we safe? There is an improved remediation method. Then we will open what we call a Synk Update PR and say, "Hey, here's a fix," even if you patched the previous ones. So even if you previously patched. Now there's an upgrade. We think an upgrade is better than a patch. We'll open a PR with that update. So we really try to, as much as we can, kind of have your back.

[00:37:46] JM: Those patches are proprietary, right?

[00:37:49] GP: They're open source. They're available. You can see them, and they're available at the licensing that matches the library –

[00:37:55] JM: Okay. Yeah, I was going to say. That's cool. If you can make a patch, why couldn't you just submit that to the GitHub repo?

[00:38:06] GP: So to clarify a little bit, right? Patches that we do really fix the dependency chain problem. So it's because B already has a fix. It doesn't need anybody to submit anything to its

feed. It's A that doesn't have an upgrade, that doesn't have those – We considered it, but starting to suggest to A that they would upgrade this version of B. We tried it – Open source contribution is this finicky area. If you do it at one place, if I went to A now and I said, “Hey, you’re using this vulnerable version of B. You should upgrade.” It will be well-received.

If I found 10,000 repositories that use a vulnerable version of B and I opened the PR to either one of them, I'd be spamming. I'd be considered a bot that is spamming the ecosystem. So very finicky. I actually agree with both sentiments, but it's a little awkward. So we had these – This is early days. We talked about it early on with GiHub a little bit, and it’s just basically the norms that were created in the open source sphere. So if you're A, what you should do and what many do do is they use Snyk to secure their project and then they will get the notifications. But if you haven't opted in to do so, we’re not just going to open unilaterally PR's to fix vulnerabilities. It might be usable.

[SPONSOR MESSAGE]

[00:39:33] JM: When I’m building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We’ve also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:41:22] JM: What are the expansion opportunities? So you mentioned compliance. You mentioned container security. Tell me about those adjacencies.

[00:41:39] GP: So the business in general, like the way we make money, is as organizations bias for use kind of across the org, right? Generally, people purchase Snyk for the sort of more modern parts of their stack. Then we also support a lot of the sort of basically your use of open source all the way back. So what pays the bills today is primarily the open side, although about 40% of those who buy our sort of Snyk open source to secure their open source use also by the container add-on, which now is, as I said, stand alone as well.

So both those are doing really well. Just for context, we 7X revenue last year. We're going to 4X revenue this year. We're sort of on a pretty hyper growth rate. At the moment, we're 70 people at the beginning of the year. We're just over 200 now. So business is good on this front. Where we see kind of our vision and our destination is to continue tackling more of these security problems that have now landed at the developers lap.

So as the scope of the application grows and more and more infrastructure moves into it, more and more changes happened to software development. The only way to scale that is to get the developer to use it, and developers don't want 17 different tools. Nobody really wants 17 different tools tackling this perspective and that perspective. Also, they're all very intertwined. You take this app, your code. You put in a container. You use these libraries. You configure it. So each of these things that I mentioned might have security kind of flaws in it, but they're very interrelated.

So really what rebuilding and where our business grows is all around tackling more security threats, but we're doing it – We're pacing ourselves to ensure that at any given step we maintain a really good developer experience.

I'll give you an example of this for containers. So containers, as I said, we took this like dev mindsets to them. So for example, we sort of sat down and said, "Well, what's a good developer experience for securing containers?" Well, for instance, when I find a vulnerability in a container, a core consideration is did I introduce that vulnerability in a Docker file or is it in the base image?

So the product really bifurcates on those two as, "Well, if I found a vulnerability in the base image, I'm not going to tell you to upgrade library so and so to this version, or like the binary so and so, because that's not the way you want to fix it." I'm going to tell you, "You should change the base image to this." If I found a vulnerability that's in the Docker file, I'm going to tell you, "Well, you installed Curl, and that installed this library, and that installed that library, and that library is vulnerable. I'll give you a context," and then I'll tell you that Curl was installed from this line in your Docker file, because that's what you want to know as a developer.

So all of those are – They seem obvious when I say them, but, generally, security tools, definitely container security tools, have come from the governance mindset. So they're saying, "Hey, here's an image. Here's a bunch of files on it. They're vulnerable." "Hey, here's a bunch of vulnerabilities. Go fish," instead of figure out how to handle it. So that's an example of how we tackled it in containers, and we're going to continue tackling it in more kind of sort of a larger scope of responsibilities for developers tackling more and more security threats.

[00:44:47] JM: What about by compliance?

[00:44:49] GP: Compliance? We also tackle license compliance as part of it, so are using components. It's very common that it's being handled together sort of side-by-side. I sort of fully say that we are a security company that also does license. There are a lot of companies in this space that are license companies that also do compliance.

I think we find ourselves in kind of the current ecosystem that we have, again, the most developer-friendly license compliance solution out there. But we really are – Our core, our kind of heart is around security concerns, and we just – We listen to customers, and our customers have told us we expect this product to also do license compliance. When we do something, we do it well. We don't really do it is a side ability. So we build good license compliance. We also do something called kind of dependency health to let you know if this library is like you're using this components and it's stale. It's just like this repository. It's not changing forever.

A more recent ability that we actually kind of quite keen on and actually not 100% sure it's been sort of fully public to launch, but it's been on for a bunch of months now through future flags, is dependency upgrade management, so, "Hey, you're using libraries."

[00:45:58] JM: Software Engineering Daily exclusive.

[00:45:59] GP: Yeah. Indeed. Indeed.

[00:46:02] JM: You heard it here first ladies and gentlemen.

[00:46:04] GP: Again, sort of we see it as a part of – It all comes down to the developer experience. What do you want to be embedded and what is it that you want to run with? There's lots to do. I don't think we're going to be bored anytime soon.

[00:46:17] JM: How hard is it to build license compliance?

[00:46:21] GP: So from a dependency perspective, it again comes back to these two bits, right? First, we have to figure out which components you're using. So we're already doing that. So adding license compliance, that part wasn't additional work. We already know what you're using. The second bit is to create a database of licenses. The first pass of that, pretty easy. You do pass. You get some licenses, and then really you start evolving it from there, because licenses are not always as explicitly stated. Sometimes components have multiple licenses.

Funny story, is we actually had – We have sometimes fix PRs for license compliance, and the way we find out is we forgot to turn off fix PRs when we added license compliance. It turns out

in the ecosystem, sometimes a license or sort of a library will come out with either no license or some strict license. Then as it gets popular, with pressure, with peer pressure from the ecosystem, they'll release a version of it that is more permissively licensed.

So basically, a lot of these mechanisms are the same as these dependency components. The true focus complexity is around building that database and just understanding the subtleties overtime. What we don't do, which is a different set of a threshold is like snippets, like, "Hey, did you copy+paste something from Stack Overflow into code?" Therefore you might be in violation.

I would say today we rarely hear like only the most strict organizations really fuzz about that. Today, we're sort of living a little bit more in an era of packaged software. While you still copy+paste, the much more prevalent concern around compliance is at a component level.

[00:47:57] JM: The engineering of the vulnerability database and the scanning infrastructure, I'm a little bit curious about – Can you just tell me a little bit about your software infrastructure. Give me the high- level software architecture for how Synk works.

[00:48:15] GP: For that back office? For the threat intel system?

[00:48:18] JM: Just on your server side? What infrastructure are you using? What cloud providers are you using? What database do you actually use for this vulnerability scanning database and so on?

[00:48:28] GP: Sure. There are a whole bunch of this. So Snyk itself –

[00:48:31] JM: You can time box to whatever.

[00:48:33] GP: Yeah. Well, Snyk itself runs – We're primarily a SaaS offering. So it's primarily hosted. We run most of our systems today on Google Cloud and a portion of it on AWS. We're a node shop primarily. At this point, especially as we tackle multiple languages, we have different pieces of code that are in native-language. Sometimes if you want to understand Python dependencies, you're better off writing a microservice in Python because of some access, any kind of language subtleties.

But the majority of our system is written in node, which was also the first ecosystem that we supported. So it was kind of related. In terms of like that stack, we're sort of a Kubernetes job. Do we use a variety of – Kind of our own component is Auth0 for authentication. We use Logz.io for logging. So we use a bunch of those sort of services around us.

[00:49:28] JM: Israeli company.

[00:49:29] GP: Israeli. Yeah. Some Israeli kind of [inaudible 00:49:31] over there. Auth0 isn't Israeli. We love them as well. We use HackerOne for our bag bounty. So we kind of leverage the ecosystem as well, and it's actually kind of fun when many of these customers are – Many of these companies are also our customers. So that's also set of a nice tradeoff.

[00:49:46] JM: Those circular dependencies are hilarious. I interviewed LightStep about distributed tracing and they're like – He gave an example and he was talking about Lyft, which used Twilio, which uses LightStep, and he was talking about taking a Lyft himself. We do live in incestuous software ecosystem.

[00:50:07] JM: We do. Yeah. Yeah, we make the most of each other. So that works.

[00:50:10] GP: It's funny, because in the 90s it was like, "Oh! Startups selling to startups. That going to lead to a big bubble." Then now startups selling startups. I don't think it's as much of a bubble. It's just more of a reality.

[00:50:24] GP: Well, I think we're all kind of early adapters because we're just sort of natural risk takers and the sort of leap of faith and are also able to deal with some complexity, while larger organizations often times have more constraints. You do see a lot of larger organizations that are explicitly working on being able to take risks and sort of use innovative startups on it.

[00:50:48] JM: I mean, that's the heart of KubeCon. It's one of the – I mean, it's called KubeCon, but that really is one of the trends, is like big companies trying to adapt technology faster.

[00:51:02] GP: Yeah. That is existential. Otherwise, they're busy just going to grow stale and die, or they'll be secure all the way to bankruptcy, right? They won't take any risks. But the world would just blow past them.

[00:51:16] JM: Not to change the subject, but tell me about the modern enterprise buyer, because it seems like that enterprise buyer is changing. It's becoming more open to taking risks to adapting technologies more aggressively.

[00:51:31] GP: Yeah. So multiple variance to it. But one aspect that's happening is the whole bottom-up and sort of empower developers. So what you're saying is certain groups. Well, before, you might've had some CIO that sort of makes some top-down decisions and everybody needs to sort of fall in line. What you see now is you're seeing a lot more decision power within groups. So then much of the organization might still remain conservative, but a slice. One group or one team is more empowered to make a different decision. Then if they're proving successful, that might sleep over to another one. Now it's proven to work within sort of this big banks ecosystem or something and they'll grow.

For us, all of our seven-figure deals today are expansions and they've all started and it might be an acquisition, they got acquired into a company, right? It might just be a modernization team. So I think that's maybe – In context of KubeCon, that's probably the most powerful element. Beyond that, there are companies that explicitly put like these innovation agents. Actual people whose entire job is to connect F500 companies to the startup ecosystem and they kind of act as a go-between where they understand the complexities of their businesses and how something can be adapted and who might be interesting with of interesting for and working with sort of the VCs or other sort of startup ecosystems to identify kind of innovative companies that might have something to offer.

I think that latter is especially impactful I think in the world of outside of dev, because maybe those teams are not quite as empowered today. But in the world of dev, I think it's more around kind of those deeds. We see that all the time. It's sort of that bottom-up motion and comes hand-in-hand with this sort of DevOps and desire for speed. So if you don't empower those teams, you're not going to move faster and you're going to be left behind.

[00:53:24] JM: You were CTO of Akamai for a while. How does starting your own company compare to working at a large company in a senior role?

[00:53:34] GP: Yeah. I got into Akamai because they acquired my startup, right? I founded a company called Blaze, which did frontend optimization and it was acquired into Akamai. I think when got acquired – So I've been acquired twice. Once into IBM. This was Watchfire, and IBM acquired us, and then Akamai acquired Blaze. It really boils down to the appetite of the buyer coming into it.

So being an exec at a large company as a whole is about scale. You have to consider that things are going to take longer. But when you get them to go, then you're kind of mobilizing a much greater force. In Akamai, if you change Akamai, you change the Internet. So in Akamai, I had the pleasure of working on HTTP 2 on sort of driving certificate transparency in promoting TLS on internally building kind of connecting tech talks between the engineering teams in across Akamai to help foster a collaboration and engaging with the performance community, the web performance community.

So there's all these changes that you need to acknowledge they take a long time. When you get acquired into those companies, it's oftentimes you get some extra benefit of the doubt, know because you're being acquired from the outside. The company has is literally decided to sort of spend millions on getting much more the new, but in part, getting you as well. Therefore, you get a bit more credit. After a while, it wears off. You have to sort of stand on your own two legs. But at the beginning, it helps you mobilize change.

I don't think I – I like both. I like challenge and I like growth, and I think at Akamai there are some amazing people, and indeed we've managed to do some great things. At the end of the day, I like creating. So I got the itch and left to found Snyk.

[00:55:26] JM: Congratulations, by the way. So you founded 3 companies at this point.

[00:55:29] GP: No. Watchfire wasn't mine. I worked for a sanctum the got acquired by Watchfire that got acquired by IBM. Then I founded Blaze, and then acquired by Akamai.

[00:55:36] JM: That's cool. You've also founded a podcast, Secure Developer.

[00:55:40] GP: Indeed.

[00:55:41] JM: So my experience of podcasting is that these emergent themes develop. I mean, there're a lot of cool things about podcasting. One thing that's cool is you do enough podcasts. You see emergent themes. That totally takes you by surprise. Is there anything that stands out? So The Secure Developer, by the way, you interview CSOs, or CISOs, chief information security officers, security people. What themes have stood out in conducting those interviews?

[00:56:12] GP: So very much, like a lot of insights come out of those. I think that's what I love about it really. Sort of getting perspectives and getting kind of smart people to sort of share their experiences and their views.

I would say the primary theme is the acknowledgment in the world of security that security needs to change as well. When you talk about building security into development, sort of the first objection that comes out is, "Hey, developers are not going to embrace security. They don't care about security. Whatever it is." I think we've successfully disproved that.

But what is not discussed as much as how the security teams need to change, and security teams have very much been defaulting to this naysayer, to these cynical entities that are all about governance and like putting, you might say guardrails, but oftentimes blockers, roadblocks in the paths. That indeed doesn't work in the new ecosystem. They've generally been a source of like old-school security teams. They think of themselves as this source of authority, of power, and they hold on to that power.

What I observed when I talk to different teams and aligns with their different practices is they increasingly think of them as a service provider. They think of themselves as they're there to help developers do the job right. A lot less adversarial. A lot less negative. You see the actual teams the makeup of the teams changing. They hire people more from a software engineering background. Not necessarily sort of SysAdmin or sort of It or just pure security and risk background, and they build software. They build – Whether it's libraries for developers to use or like internal services for them to run.

So all of those perspectives, having a positive mindset, you see a lot of – I was sort of joking with the CISO of one medical about hoodie driven security. There's a bunch of these. Like one medical has – They have a security hoody that they give to developers that do a good job. I think Segment have like these special stickers that you get for your laptop if you kind of pass some security training. I think Optimize, they also have this like security hero T-shirts that they give out.

This notion of like thinking about how to incentivize, how to not just sort of bash someone on the head if they have done something wrong, but also celebrate them when they do something right. So all of that change in attitude in the security teams is something that I don't think I understood or sort of appreciate as much, and it's very much a theme and also almost like a very clear delineator when you talk to someone. If they're just talking about fixing technology and they talk about them. How should developers change? Are they looking internally? Looking in the mirror and saying, "How should we change? How should we adapt?" I think those are the ones that are more successful.

[00:58:57] JM: Well, Anchor, if you will check out that podcast. I have not gotten a chance to listen to it, because you told me about it one minute, or an hour ago, or whatever. But I'll ultimately check it out.

[00:59:06] GP: Try it out. The Secure Developer.

[00:59:07] JM: Last question. If you weren't building Snyk, what company would you be building?

[00:59:13] GP: Ooh! I had the option and I chose to build Snyk. It's hard. Snyk is so sort of ingrained in my being at the moment. I think I guess what I can answer is like what would I do next. Snyk is probably a very long journey still ahead, but I think for me my passion is around – I'd like to do something that's more about sort of social change, and my passion is about education. I feel education is – Our core approach to education is broken.

This sort of expectation that at every geographic location, you would have a decent distribution of people that are good at teaching and good at every one of the subjects we want to teach that are available to teach kind of locally. It's just fundamentally flawed. That's just not can happen, and that creates – Top that with sort of lack of government investments and all that, but you really get to a place that is not exciting at all.

So I am a firm believer in technology, and I think you can fix a lot of that with kind of the right technological solution. So I would say my next investment, I expect, or kind of the place post-Synk, whenever that may be that I'll invest my time is probably going to be about helping tackle that problem.

[01:00:16] JM: Guy Podjarny, thanks for coming on the show.

[01:00:18] GP: Thanks for having me.

[END Of INTERVIEW]

[01:00:27] JM: If you want to extract value from your data, it can be difficult especially for nontechnical, non-analyst users. As software builders, you have this unique opportunity to unlock the value of your data to users through your product or your service.

Jaspersoft offers embeddable reports, dashboards and data visualizations that developers love. Give your users intuitive access to data in the ideal place for them to take action within your application. To check out a sample application with embedded analytics, go to softwareengineeringdaily.com/jaspersoft. You can find out how easy it is to embed reporting and analytics into your application. Jaspersoft is great for admin dashboards or for helping your customers make data-driven decisions within your product, because it's not just your company that wants analytics. It's also your customers.

In an upcoming episode of Software Engineering Daily, we will talk to TIBCO about visualizing data inside apps based on modern frontend libraries like React, Angular, and VueJS. In the meantime, check out Jaspersoft for yourself at softwareengineering.com/jaspersoft.

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

[END]