EPISODE 1506

[INTRODUCTION]

[00:00:00] ANNOUNCER: This episode is hosted by Sean Falconer. Sean has a PhD in computer science, was a postdoctoral student at Stanford's Medical School, and is an ex-Googler and startup founder, now serving as head of Developer Relations at Skyflow, an architectural solution for data privacy. Sean has published works covering a wide range of topics from information visualization, quantum computing, developer experience to data privacy. You can find more of his work by following him on Twitter, @seanfalconer.

[00:00:35] ANNOUNCER: Cloud computing provides tools storage servers and software products through the internet. Securing these resources is a constant process for companies deploying new code to their cloud environments. It's easy to overlook security flaws because company applications are very complex, and many people work together to develop them.

Wise Labs, for example, had millions of users' data stolen due to a mistake by a single employee. The company Bridgecrew is a cloud security platform helping to prevent mistakes like that from happening. Bridgecrew integrates into developer workloads to automatically find infrastructure errors in cloud accounts, workloads, and infrastructure as code. Their platform also monitors code reviews and builds pipelines to prevent errors from being deployed into production. If an error is found, then Bridgecrew Software reverts that code back to its last known correct state.

In today's episode, we talk to Guy Eisenkot, VP of product and co-founder at Bridgecrew. Guy previously worked as a principal product manager at RSA Security and as a product manager at Fortscale before that. We discuss infrastructure as code, DevSecOps, cloud security, software supply chain and composition analysis.

[INTERVIEW]

[00:01:50] SF: Guy, welcome to the show.

[00:01:51] GE: Hey, Sean, happy to be back.

[00:01:54] SF: Great. Yeah, it's been a while.

[00:01:55] GE: It has been a couple of years.

[00:01:57] SF: Yeah. So, I think given that it's been several years, and I'm sure a ton of things have changed on your end. I think, to kick things off, can you start by introducing yourself and maybe share a little bit about your backstory?

[00:02:09] GE: Yes, absolutely. So, hi. My name is Guy Eisenkot. I was up until very recently, the co-founder and the VP of Products for Bridgecrew, and we'll mention Bridgecre in a sec. We were acquired by Palo Alto Networks in 2021, and ever since I lead product for Prisma Cloud in a subcategory called Cloud Security. Prior to founding Bridgecrew, I've built products in the machine learning space and analytics as well. Generally, I can say, that I've been trying to solve human problems with data as long as I can remember. So, I'm firmly not a developer. I actually majored in history and economics, but I've been building foundational software for the past 10 years.

[00:02:50] SF: Awesome. Yeah. Thanks for that. I think you have a really interesting depth of background across a lot of different parts, both from like an engineering standpoint, as well as in product. So, I think this will be a fascinating conversation today.

[00:03:03] GE: Yes, looking forward to it.

[00:03:05] SF: Yeah, so Bridgecrew embeds security directly into the development lifecycle, across a few different areas to make it easy for, essentially developers, to secure infrastructure as they build it. I think in order to talk about how you do some of these types of things, and some of the products that you offer, I think it makes sense to maybe start out by talking about if a structure as code, for anyone that might be a little bit unfamiliar with it. Infrastructure as code, I think, it's been around for a while, but the growth and interest, both in, I think DevOps and the movement, the cloud. It feels like awareness and excitement in this field is really on a fast growth path. Companies are starting to utilize frameworks like Terraform and CloudFormation, to build change and version cloud and on-prem resources. What do you think this sort of shift has happened? And what value do these frameworks provide to business?

[00:03:55] GE: I'll give you a bit of a backstory. One of my co-founders, actually, the technical one, taught me something very profound in our previous company. And it's been like a light tower for me, ever since. It's something foundational about software development, which is building good new products, not necessarily sustaining the existing one. Actually, it means to be constantly in a state of migration, and that's a really deep way to understand how software engineering has changed, and what infrastructure as code and Terraform had to do with it.

So, when you think about the adoption of infrastructure as code, it's really a brainchild of that mentality. If you want to make the absolute most of a modern application stack and use modern development practices that allow you to build very fast and ship very fast, you eventually need to be able to adapt and change effortlessly. What Terraform allows and infrastructure as code framework is a decent substitute to that, Terraform just being community projects. It manifests that even more, is that it unlocks that for a growing part of the application stack. It reduces huge friction by obstructing something that we all use today when we build a modern web application, which is cloud provider API's. By providing us a single configuration surface through either a new programming language or an existing programming language that invokes infrastructure as code on the back end, it just goes way beyond what you need to be able to run a modern web application. So, to sum that up, I think for a growing business, the utilization of infrastructures as code is almost a mandatory prerequisite for building something new.

[00:05:37] SF: Yeah, I love that idea of the constant state of migration. One of the great things, I think, about some of the specific cloud providers like AWS, Google Cloud, Azure, and so on, is that they do provide these fairly easy dashboards where like a non-expert in infrastructure, can create cloud services, adjust permissions, provision servers, and so on, with just a few button clicks. Essentially, does infrastructure as code need to live separately from that? Can these things coexist? Or is this something where if you're choosing essentially infrastructure as code, then you don't want to you rely on the UI tools, because things are essentially going to get out of sync?

[00:06:17] GE: That's actually one of the most beautiful aspects of modern infrastructure as code, is that it can be both. It can be a path to migrate for an existing legacy infrastructure, to a net new stack. It can allow you to be a fully blank canvas where you can build an application from scratch. For us at Bridgecrew, I can say that multiple life cycles of development throughout the product history, we've

essentially built large portions of it from scratch after improvements and enhancements were introduced to then AWS services that we use, just in ways that we can simplify the things that we did previously.

So, I think the benefit is not only the fact that you can use this very simple and intuitive programming language and not whatever, the dashboards in the UI, it also caters to what you've already done previously. It can work on top of it. It can work aside it, and it can fully replace the way that you've configured and provisioned infrastructure previously.

[00:07:15] SF: So honestly, like companies like Google and Netflix, Facebook, other famous technology companies, and even what we consider a traditional tech startup, are using a lot of these techniques around infrastructure as code. But is this something that you're also seeing, a nontraditional technology company adopt as well, like a company like Walmart or Target. They probably have fairly large tech department, but are exactly what you think of, when you think of technology innovators. Are these types of companies also migrating in this way?

[00:07:43] GE: A large portion of them are. We're seeing some – so, right now, I think I have an interesting preview now that we're part of this huge global enterprise called Palo Alto Networks. It secures probably up to, more than 70,000 customers globally. Our division handles a subset of them. And you can see two classes of engineering teams. One is the classic, you mentioned, the born in the cloud, the fast facing, and actually, the easy side, people who understand that this is just the way that they can utilize the most of their cloud infrastructure. And then there's another camp, which is the digital transformation camp. Actually, these are some of the biggest names and brands in the world, where there's a group of people that understands the value, but then there's another group of people that's very dedicated to continuing to serve what I'll call the traditional, the legacy infrastructure.

So, for those digital transformation companies, they're even more excited about infrastructure as code, because it just allows them to reduce some of the friction and get more people on board and more people excited to adopt some of these development best practices.

[00:08:48] SF: Right. Yeah, that makes sense. So, when we talk about starting to think about like security, when it comes to infrastructure as code, what are some of the common mistakes that like a team makes when managing security for frameworks like Terraform, or CloudFormation or others?

[00:09:03] GE: I think one common misconception is not to realize that infrastructure as code is just like any other programming languages. T follows the same design patterns and effectively carries the same types of risks. On one hand, it offers structures that allow developers to define what they want to do, which is exactly the same if you compare it to a Node or a Java application. And they allow you to build individual instances of artifacts, usually pretty straightforward, with no mirrors. 10 lines of code, you can now build scalable databases and a global infrastructure and that's pretty cool.

From a security perspective, the main challenge is defining how those instances of infrastructure talk to each other, and how those connecting to planes that are usually the ones where most of the mistakes happen. One is the network plane, the other is the identity plan. Not to say one is more complex or interesting than the other. That actually varies between cloud providers. But that's definitely something that's been a growing concern for a large portion of it. Terraform, CloudFormation, developer communities, since cloud providers have provided a lot of good defaults to get your cloud infrastructure up and running. Those good defaults, make things talk to each other and connect pretty regularly. But they're not really designed to be optimized for security concerns.

So, unless you're building a payment application, and you have to adhere to very strict segmentation rules and laws, most of us that are building enterprise software, have to kind of work against intuition and understand and get familiar with the limitations of each and every type of cloud service. And then, when we do our job well, we get to an end state where identity and networking are defined to serve the business and make the application work, but they're also serving the purpose of making sure that my application that actually runs on the internet doesn't have things that are exposed to it and could potentially become compromised by that.

[00:10:53] SF: So, in these situations where companies are using infrastructure as code, something like Terraform, who is typically responsible within that organization for these security concerns? Is that the security team or is that going to be shifted to like the DevOps team or even potentially, with the application engineering team?

[00:11:13] GE: I've been fascinated by that question for the past three years, ever since founding Bridgecrew. One of our early, but very profound experiences going out to the market and trying to educate ourselves about how traditional security teams, the one that do corporate security are now ingesting this new task of implementing security controls, is learning that not only is there a knowledge

gap, but there's a pretty substantial access gap. SOX, three years ago, four years ago, didn't have access keys, and good familiarity with the ways you access a cloud console, to be able to even collect logs or iterate after the fact.

So, it's interesting to see that what was supposed to be a security concern, turned out to be something that's concerning to different types of organizations. Development organization has some roles and responsibilities, then security has some roles and responsibilities. And the emergence of DevOps and DevSecOps has created additional people in the organization that are now interested in making sure that when infrastructure is built, it's done through a CI/CD pipeline, and that there's proper controls validating the changes that are being made.

Long story short, it's a lot of different people. I had, up until, I don't know, a year and a half ago, probably pre-COVID, almost two and a half years, I had this ambition or hope that we're starting to see more centralized infrastructure and platform engineering group form, where the understanding is that this is such an important job that organizations will be form dedicated organizations within their engineering groups that are in charge of secure packaging, and distribution of premade secure defaults and templates. And that's something we've seen in the financial sector and somewhat in the IT sector as well. I thought this will just spread like wildfire. But I think for a variety of reasons, that process actually slowed down. We've been talking to our peers at AWS and Google Cloud to see what they think of this, and they're also agreeing that instead of infrastructure as code, security becoming being this own thing that small group of people is very interested in and excited about. It's now become much more apparent that larger and less dedicated engineers are getting tasked with working with these types of frameworks.

[00:13:29] SF: What's the danger of tasking those teams with this responsibility?

[00:13:34] GE: There's a few. I think the biggest one would be how easy it is to make a mistake. So, even before we go to the specific types of use cases, there's something about cloud infrastructure that breaks down a lot of the organizational barriers that developers had previously. And then think about how infrastructure as code which is, again, 10 lines to deploy a multi-region instance of a database that can now store personal information of your customers. This is literally 10 lines of code, with probably 30 or 40 different types of attributes you can toggle around with. It's really common to see that those 10 lines of code don't necessarily contain the, I don't know, five, six, seven, complete best practices for

deploying that set of infrastructure. Essentially, if there's no guardrails along the way, we see developers that are even security aware and not necessarily aware of all the different knobs and switches within a specific type of implementation that the cloud provider has exposed for them.

[00:14:34] SF: So, this is probably, I think, a good time to start to talk about Bridgecrew and some of the products and services that you offer. How does Bridgecrew help prevent people from making some of this common infrastructure as code, security mistakes that you've mentioned?

[00:14:49] GE: Bridgecrew essentially, pioneered what I'll call crowdsourcing and standardizing infrastructure as code security. So, when we founded the company in 2019, we saw companies of all sizes, struggled to implement security controls for both, centralized governance, security and compliance reasons into their application development. And essentially, we've built software, open and commercial to educate the market on its importance. One of the projects that I'm most proud of is not our most popular one. It's our training project. It's called TerraGoat, and it allows people to download a local instance of infrastructure as code configurations, from different types of frameworks. And then, to see if their existing pipelines will prevent them from pushing that code into production. So, that's something we believed in fairly early on, and we've kind of built our entire vision on how we can now educate them on the security market, and then educate developers on just using the right tools to be able to mitigate as much as possible and then prevent as much as possible some of these mistakes forever getting into production.

[00:15:56] SF: You originally founded our company in 2019, which is not that long ago, but even in these few years, have you seen sort of a change in the understanding and awareness of these potential security issues when it comes to infrastructure as code deployments?

[00:16:13] GE: Oh, absolutely. I think not even in one specific instance. I think, there were a few waves. The first wave was around that time when the company just got off the ground, and it wasn't just us, it was a few other similar open source projects that have made it and this is where the cloud security community gets a lot of credit for not only being able to – we do great conferences, but we also build a lot of internal tools and we open sourced them. And there's just this very dedicated group of a few thousand developers worldwide that have encountered problems since the early 2010s. And instead of writing a blog about it, they just build tools that solve them. And we were very inspired by those tools. I think, starting 2019, those tools have really started to catch on and standardize, just the

sheer understanding of what a misconfiguration in the cloud is, and what are the best ways to mitigate it. It's not just creating alerts, or building JIRA tickets, that there's an opportunity here to build a very efficient and automatic workflow to prevent developers from ever pushing that bad code into production.

[00:17:18] SF: Yeah, I love that idea of really baking the best practices. So essentially, mistakes don't happen.

[00:17:24] GE: Exactly. One last comment on that, is I think there was a second wave about a year and a half ago, that was sparked by a few very high-profile disclosures of misconfigurations that were conducted by the cloud providers themselves. So, this is where researchers from Palo Alto, from a few other very similar product research groups, like Wizard, Orca, and LightStream, interestingly enough, some of them operate from the same building blocks as bridge crew, have been able to identify pretty staggering instances where AWS, Google Cloud and Azure have failed to correctly configure different types of infrastructure, and open the cloud services themselves for abuse.

In those cases, infrastructure as code would not have necessarily been the optimal solution. But it did do a lot to raise the awareness that infrastructure is this now new front in cybersecurity, and as a up and coming topic, it does have its standards and best practices that help you prevent some of the exposure that comes in from that type of risk as well.

[00:18:28] SF: And Checkov is Bridgecrew's open source policy as code tool that scans things like my Terraform template for the security issues. So, can you walk me through like how I would go about using Checkov and integrating that into my infrastructure as code deployment? And what are the outcomes? Is this is like a pass-fail, like a unit test? Or it sounds like this is probably something a little bit more nuanced than that.

[00:18:52] GE: It's both. So, the nice thing about Checkov is that it does follow two design patterns. One is that it really is constructed like a testing tool on one side. But on the other, it's built like a lot of the cloud terminals and CLI tools we use to provision infrastructure. So, it brings in some of the logic and design principles that you've seen in tools like Terraform, Packer, Salt, Ansible, Puppet, which our users are very familiar with. And then the other hand, they just play very nicely with tools like CI/CD tools like, CircleCI and Jenkins and Azure DevOps and AWS's code build, and just the ability to run as an ephemeral job that looks at your code, and provides a very straightforward set of results that, as you

say, a pass-fail in some context for the developer to untangle, but in an interface and in a UI that they understand.

So generally, there's two operational patterns. One is just the baseline. So, you can either download Checkov locally or run it from any Docker, scan your code, and get a pretty robust report about every imaginable misconfigurations you can think of. So, it started with infrastructures code, but it does image analysis and it does hand-on abilities in open source package managers, and it finds secret, and it's gone well beyond that. But you can do that against your existing codebase, you get a nice report, and you can essentially get a good grip on what that repository is potentially in risk of. And then the more common operational flow is to include Checkov in a CI pipeline. You can include it in, let's say Jenkins and use an example of a new choreograph synchronized job in Jenkins that run, usually, in a staging environment, pre-deployed, post deployed, it doesn't really matter. And it just performs a set of checks on all of the configured and changed code. So, instead of getting that full report and all your existing repo, you get the results only to what's changed in this specific run. And then whether that's be the developer that now pushes this CI job or tries to push it a side branch into the main branch, depending on the branching strategy, they get this report as they evaluate other unit tests and outcomes for their CI/CD system, and gives them as much context as they need whether to ignore the problem or to solve it.

[00:21:08] SF: So, how much, I guess knowledge and experience does a typical user of Checkov need in order to use the tool properly?

[00:21:18] GE: That's the beauty of it. There are two types of Checkov users. There's the, you can call the evangelist and then there's a passive user. The evangelist is usually the person that brings this tool into the pipeline. This is usually someone from either a security or DevOps background. They'll need to have some basic familiarity with infrastructure as code to be convinced that the result is good enough to be included in the standard set of practice, and also in how the pipelines are configured in order to make sure that outputs are printed in the right steps and the reports are viewed by the right people.

Passive user, which I think is the more interesting of the two, really doesn't need to know anything about infrastructure as code, just like they don't need to know everything about everything else. They get tested during CI, and that's what's nice about that second workflow. So, after that initial baseline has happened, any developer that understands programming can then see a printed version of a

Checkov result, it will then highlight what security control has failed. It will provide a link to a guideline, a public documentation that we write, and open source as part of checkout that includes the logic of the tests, why it failed, and potential remediations. And then they can decide, and effectively they see a printout of the code snippet that has been misconfigured. And then taking those three aspects into account, they can decide whether to resolve using the recommendation that we provided, seek out another recommendation on Google or Stack Overflow or disregard the result, print out a skip comment, and continue to the next test.

[00:22:52] SF: Okay. So, if I understand that correctly, basically, there's probably someone like a DevOps person that's doing the initial sort of integration setup. And then as that's integrated in the CI/CD, all the application engineers are essentially going to be utilizing that framework because it's baked into the CI/CD, and that'll create this report where they can catch these errors and make adjustments as needed. Is that correct?

[00:23:16] GE: That's accurate. Yes.

[00:23:18] SF: And then, in terms of best practices, are there already best practices for common infrastructure combinations and challenges baked into the system? I imagine that there's so many different ways that you can create these different cloud resources and combine them in different ways and have different levels of access. How is that sort of managed? How do you keep essentially the system up to date with these different combinations and the best practices around them?

[00:23:46] GE: So, there's two parts of that question. One is, what are we actually enforcing? That's actually based on common frameworks that we, as an industry, have developed over the past, I would say 10 years, stretching that back, as far as virtualization standardization has begun. So, if you think about CIS benchmarks, and there's a bunch of them, but different types of bodies that we have decided to trust have given us this initial checklist of items we should look at before provisioning new infrastructure. And cloud providers themselves have gone a long way in providing frameworks to make decisions about how infrastructure should eventually be configured when it goes to production, and it follows a few basic patterns. They would usually look at things like making sure that again, networking and identity usually come up first. This is networking just to prevent that your infrastructure isn't exposed to the public internet, identity that people shouldn't be able to access your services are able to

access. And then there's a long tail of additional verifications that a well-architected manifest should follow.

These include things like ensuring that encryption is properly configured for databases or data in motion infrastructure like Cues and Kafkas, and these sorts of services. We want to make sure that logging is used consistently. Each service now packs on its own logging sub tools. So, sometimes we need to have additional checks that look to see that the required logging controls are in place, monitoring on top of those logs, that's also something that we define in infrastructure as code. And a few more, one other big one is the correct use of secrets, environment variables, user and passwords, right? These infrastructure components eventually need to be communicating with each other. They need to share secrets to make sure that that's done in a way that doesn't expose those secrets, once those infrastructure components are spun up.

So, there are like six or seven examples of the types of failed issues that a Terraform file going through Checkov might look at. Your second question, which was also interesting was how do we make sure that we continue to keep this up to date? When we started Checkov three years ago, we had this expectation that we develop or build some in house research and build out some expertise around what should be these controls that are implemented based on customer base, and discussions with the community, et cetera. Something very different happened. Instead of us building that, we built something like the first 100 or 200 policies. The next 1,800 were vastly contributed by the community. We just found that, people are building in such different manners using existing infrastructure, that our ability to track, what is it, like 10,000 different discrete cloud provider API's, and about 500 different types of cloud provider services between Amazon, Google and Microsoft, just an impossible task for a single team. So, that's the part that was crowd sourced, and most of the policies are now managed and tested through our community validation process that happens on a public Git repo.

[00:26:55] SF: Yeah, it's awesome. I think one thing that I did want to mention is that you have, from the beginning, invested in open sourcing some of the tools and technologies that you built. Why was essentially making those tools open source important to you and the rest of the founding team at Bridgecrew?

[00:27:16] GE: Right. So, for two main reasons, one was that we have a very deep and almost philosophical relationship with open source. We've used open source in our previous company. We've

used open source extensively, when we build the foundations for Bridgecrew. We use cloud security tools that the community has curated and helped us both learn and cater to some of the use cases that we saw in the market. And also, our stack naturally, was built out of open source. Who can build a modern application without open source?

So when it came time to decide whether or not Checkov should be open source, we asked ourselves, what's your goal? The main or the deciding factor for open source and Checkov was an understanding that this is a problem that every developer will eventually have, and open sourcing a tool that helps them solve, it probably does the best service to educate the market and make sure that this use case gets the right attention. That was the original call. Over time, we saw that there's other benefits of building a community around an open source tool as a commercial company. And when we made the decision to join Palo Alto, we learned that not only do we have that vision, but also senior executives at this company understand that. I think that vision has been one of the main contributing factors to its success as a code scanner, and now, much more than that.

[00:28:35] SF: Yeah, it's awesome. I mean, it sounds like even from just the sheer scale of the number of ways people can configure infrastructure, having those community contributions back to Checkov, to incorporate best practices and incorporate the growth and cloud that as a single company you can never keep up with is amazing, and probably makes for an overall better product for everyone.

[00:28:55] GE: Absolutely. I can't agree more on that statement.

[00:28:58] SF: So, one new area that you're helping companies address security needs is with the software supply chain. And just so we can set appropriate context for this topic for our listeners, can you first explain what is the software supply chain?

[00:29:13] GE: I have my own definition. I think one of the problems with the emerging space in the cybersecurity market is that people create their own definitions, but I have my own. I hope it helped. But the software supply chain problem and where this is coming into the discourse is a growing understanding that this because we build infrastructure, and as we build it in a distributed fashion with global teams that are essentially located anywhere, we need to be much more conscious and aware of what dependencies are used as part of that construction process, not only one of the programming languages that we introduce, we need to have a much more consistent and better grasp of what open

source we use, what's the infrastructure that we use to deploy, to package, to compile, when to ship that infrastructure, and also to have a way to consolidate that into a single threat model that recognizes those supply chains are actually pretty powerful applications, that have access both to our source code, and also to a runtime environment. Just by that fact, they're becoming a target for external actors that think that's a good way to either steal that data, and or get access to our customer's data as well.

[00:30:24] SF: Yeah, I think one of the security areas that are sometimes often overlooked is something as simple as someone's VCS or version control system. We spent all this time thinking about code security and data security, and even infrastructure security. But essentially, if a bad actor has access to our actual source code, a lot of that security is probably mute. So, in terms of some of the things that Bridgecrew is doing around software supply chain, how does Bridgecrew help companies essentially secure the combination of their CI/CD and their version control system, and also other parts of the software supply chain?

[00:31:00] GE: This is a hard problem to solve. I won't claim that Bridgecrew solved this. But again, if I go back to Palo Alto Networks understands this. They've seen the SolarWinds attack unfold, covering it from various angles. And naturally, with large corporations like even like Microsoft getting hit, this has become center stage.

Bridgecrew's role in this play not just to look at the code, as you mentioned, to see that the code is using proper best practices is go beyond essentially, anything that you use in order to compile and ship that code. The late approach means that will allow in our open source to scan things like version control system configurations. So, if you run Checkov against your GitHub or GitLab, or Bitbucket configuration sets, we've prepackaged probably about 20 out of the box rules that make sure that you're using their security best practices, the things that they preach to, the things that they do when they build out new repositories.

And then the second part of it, it's also in Checkov is verifying that CI/CD systems are configured correctly. So, things like GitHub actions, and CircleCI and Argo CD. These are not only CI/CD systems that help you automate your delivery pipeline. But again, these are very powerful – they have very powerful applications that have a lot of access to your code and your cloud. And they also have the ability to get invoked automatically. So, just imagine what happens if someone identifies misconfigurations that can lift them. Those are pretty much the simple use cases, just ensuring the

posture of your version control, and your CI/CD systems. And then if you use the Bridgetcrew platform, you get the benefit of continuous verification of the appearance of different types of vulnerabilities within that infrastructure stack. So, we'll detect if your GitHub action is using a vulnerable version of a popular distribution of an image, right? So, you may be running Jenkins or may be running GitHub actions, but it's running on a vulnerable image. We checked that and flagged that as a potential CI/CD risk. And obviously, the combinations, right?

So, what happens if a misconfigured VCS now has access to a vulnerable CI system that's able to ship code through an open source registry that just has outdated packages that are now vulnerable to new kinds of new variants or vulnerabilities, that eventually get shipped into your cloud environment that has publicly accessible infrastructure. So, from a threat modeling, and even a threat hunting perspective, we will get some of that visibility in Bridgecrew, and you'll be able to, essentially back cast, or back trace, all of those risks to the source and hopefully even fix them with the right configuration sets.

[00:33:48] SF: You mentioned that there's like 20 best practices for – it's like baked in for securing version control system. What's an example of some of these best practices?

[00:34:00] GE: There's a few very silly, not silly, but very simple ones. I think of things like identity security. You have this for any SaaS, but some of these are just not toggled by default for some of these. So, just required to FA, that's a configuration, you need to opt in for your repository on GitHub. For companies that use SSO, so you can check if SSO is used on each and every repository. GitHub allows you to write a rule to make sure that developers can only access it using a dedicated set of IPs, if you're using something like a VPN or proxy chain to access net.

One other big recurring aspect is what's called branch protection rules. And all of the VCS providers now offer this in some variation. But what branch protection rule is essentially they are a set of controls that can be toggled on and off. Essentially, the code commit, code review and code merge processes are all governed through a set of specifically defined explicit controls. So, only collaborators with distinct access can make sure that new code is introduced into the system. Without branch protection rules, for example, for an open source repo, anyone from the internet can push new code into your repo. In closed environment, you want only, you can define something that's called the code owner that's only specific sets of people have access to specific sets of code. The simple configuration that everybody should be looking out for now, when they go into GitHub, search for branch protection rules, just toggle

them all on, 99% that's all you need. But for bigger and larger environments, you should be looking at code owners as well.

[00:35:36] SF: Does any of this additional security to like CI/CD, or the version control system, impacts performance in a meaningful way?

[00:35:44] GE: Give me a sec to think about it. Performance is a wide topic. I think there's definitely just in terms of developer velocity, yes, there's a cost to pay here. So, let's take one example, that's close to home. If you run Checkov on a large enough repository, it could take probably between 30 seconds up to probably five minutes. It's the longest run I've seen to get the results from that scan. So, if we scan hundreds of thousands of files, it could take probably a good 5, 6, 10 minutes for that scan to end and that's only what Checkov scans. Whereas, unit tests and integration tests and everything that's going on.

So, on that aspect, there's definitely a cost to pay in terms of developer efficacy, I would say. As for actual performance of the application, latency, lag runtime, not specifically. Nothing you should be concerned about, I think, that I can think of.

[00:36:34] SF: Yeah, it's a good distinction. I think one other thing that would probably be interesting for us to talk about is that you've recently started to move beyond purely securing infrastructure, but also expanding in the application security, specifically, software composition analysis. And if you're listening, and you're unfamiliar with software composition analysis, essentially the process of identifying open source software within a code base. Can you talk about why you're entering this new vertical?

[00:36:59] GE: Yeah, so I think back in 2021, we called out internally and then externally on where we publicly talk, how the lines between infrastructure and application are becoming blurred. And this notion came about with, you know, naturally, a lot of big-ticket disclosures for vulnerabilities in the supply chain, including Log4j and the month and a half, it took to really resolve and get to a proper resolution. And when you think about that blurred line between infrastructure and application, you also remember that it's not only open source that you've decided to use, and you've made a conscious decision to embrace some of that adhered risk. There's just so many dependencies today that it's really hard to keep track of, when you use managed infrastructure. So, when I use a service like EKS, that Amazon runs Kubernetes, for me, or I use ElastCache, which runs Redis, for me, or use the different flavors of

those open sources on the variety of cloud providers, I am accepting risk from those dependencies, regardless to what my software composition analysis tool is flagging up for me. And that became very apparent in the Log4j instance, when we started to discover that not only do the cloud providers run Log4j in a bunch of instances for us, they're taking a very long time to patch it.

So, I think that's sunk in, and that's around the point that we decided to take some of the internal efforts that we had around composition analysis and make some external. And we've decided to go about this in our own takers, naturally an advantage to go into software composition analysis in 2022. If you went into this, and you built an SCA, company three, four or five years ago, it was much harder for that package managers themselves. So, I think, NPM, Yarn, RubyGems, they had much less publicly accessible API's that allow you to discover those dependencies. And those API's just have become very open and very transparent. So, it's much easier for us now to not only analyze NPM, which is an NPM package, which is something that we've done for almost a year and a half now, but also to untangle all of its dependencies and build out a full-blown dependency tree using their existing registry API's. So long story short, yes, we definitely understood that infrastructure is not only vulnerable, but also a huge part of OpSec, and we've decided to build a software composition analysis that reflects the value it should provide to customers in 2022.

[00:39:31] SF: Yeah. It seems like the timing for a product like this is kind of perfect to now between instances like the Log4j instance, where I think this is going to be top of mind for a lot of people. And then, also as you mentioned, these package managers opening up these API's that actually make the technology of software composition analysis in the sense actually possible.

[00:39:50] GE: Absolutely. I think as always in cybersecurity, you have to have a combination of people understanding the risk and the other end, bad people that are utilizing the opportunities and trying to take advantage of them. Again, I think both independent researchers, as well as just attacks that were disruptive have done a very good job in advocating for why the market is right for disruption.

[00:40:14] SF: So, obviously, there's been these new product investments that you've made, and it's been a couple of years since you're on the show. One of the things that you've mentioned a couple times is that last year, you were acquired by Palo Alto Networks, and you're now part of Prism Cloud. So, how has that changed your product or areas of focus?

[00:40:33] GE: I will surprise you. I think what hasn't really changed is the roadmap. The roadmap has been pretty consistent. If I look at the one that we presented to Palo Alto back in early 2021. We've executed on most of it. It should not exceed a large part of it, just because of the added resources. There is a difference being part of Prisma Cloud and Palo Alto since eventually, it's a – I'll call it a category formation company. Palo Alto, we disrupted the firewall space 20 years ago. It's been disrupting corporate security for the past 10 years, doing probably six or seven big acquisitions. For cloud security, it's been the standard. It's acquired six companies up until – now, we're number six. And by virtue of those acquisitions, it's essentially defining what is cloud security. So, when you see the space grow, when you see what the competitors versus what the market is looking after, you should be keeping an eye on Prisma Cloud, because it is a very thoughtful group of people that's been doing cloud security for the past six or seven years in a corporate scale, and I've learned a lot just by observing the decision making and the understanding that this market just brings in huge opportunity to be a market leader.

[00:41:45] SF: That's awesome. I'm happy to hear that the acquisition from your perspective is working out. I know, acquisitions can be very challenging, and sometimes they don't always work out the way that you had hoped they would be. So, as we start to wrap up here, is there anything else that you think is important for the audience to know?

[00:42:00] GE: Let's maybe spend a minute and talk about the future. I think future is bright for application developers specifically. I think, when we talk to prospective customers, we meet people that are early in their journey into the cloud, people that are probably in there, what is it third, fourth, fifth year, building a qualification, and coming in from a technical background, not being a developer, myself, but just doing software for so long, I'm just saying, the opportunity is such a blessing for someone in this industry. And not only can you pursue a career in application security, or in application development, but you can really use the publicly available resources that are made available by companies like Bridgecrew, companies like Palo Alto, and a lot of other great contributors from Google, Microsoft, Azure, and a variety of more. You just become exposed and educate yourself and be able to build out a career in this industry, and if anyone's interested in how we've done it, and how we're planning to go forward from here, I'll be happy to have those conversations. You can reach me out on Twitter and LinkedIn, and follow up.

[00:43:02] SF: Awesome. I think that's a great place to leave it today. So, thank you, Guy, for coming on the show. I think this was a fascinating conversation, and I'm really excited to see what's next for you.

[00:43:10] GE: Thank you, Sean. Continue the great rest of your day.

[END]