Eric: [00:00:00] So in today's episode, we have Brian Behlendorf from the Open Source Security Foundation. I don't know anybody who knows. I founded a cybersecurity company, started with risk assessments of vulnerability scanning, grew it out, bought a cybersecurity company that had a platform, tried to adapt for cloud security management. But the postscript on that story is that I'm still a lawyer and one with less money. At any rate open source security always felt like a knowledge gap from my traditional cybersecurity training. And I spoke to the folks as Hyperledger and said, Hey, how can I broaden this out to, to cover more like open source specifically?

And they said, you gotta talk to Brian who's tackling this issue as a public good. And it's hard to go a single week without hearing of a crypto hack. It's hard to imagine that cybersecurity is not gonna continue to grow in importance and dominate headlines as well as the attention of regulators.

So organizations like OSSF, which operates only from grants, can begin to raise awareness and provide the tools and standards for open source [00:01:00] developers to reduce the risks associated with open source code development. So I think this is a really important podcast for anybody in the digital space and hope that it points the way to better security again, as a public good.

Lastly one term comes up a few times on a podcast in case you miss the references, it's SBOM, software bill of materials, which is an inventory of constituent components and software dependencies involved in a development and delivery of an application. It's a very important critical component of the SDLC software development lifecycle, and it provides visibility into the underpinnings of software to help organizations and users better understand what is being used and where there could be potential risks.

So anyway, with that this is a great episode. Just, take it all in it and maybe even listen to it again and again cuz I think, and check out the website too, and check out the tools that they offer. This is a great resource for anybody who's involved in open source and who isn't.

With that said, I give you Brian Behlendorf of the Open Source Security [00:02:00] Foundation.

Welcome to The Encrypted Economy, a podcast exploring the business laws regulation, security, and technologies relating to digital assets and data. I am Eric Hess, founder of Hess Legal Counsel and your host. Join me on this journey exploring the reach of these transformative technologies.

So excited to have Brian Behlendorf here from OSSF previously Executive Director of Hyperledger. Brian, welcome to the podcast.

Brian: Thank you. It's great to be here.

Eric: So, we're gonna be talking about open source today and all the work that you've been doing with the, with a relatively, a relatively new organization.

But before we get into that, why don't you give a little bit of background yourself and how you became- your road to open source security.

Brian: Sure. So let me start with what I do today, which I'm general manager for the Open Source Security Foundation, [00:03:00] which is a collection of organizations.
And frankly, all the great stuff is coming from the developers leading the project. That's it's an open source project housed within the Linux Foundation, but with its own charter, its own mission, its own budget and not just focused on Linux, it's focused on the entirety of the open.
ecosystem. I've been in the open source world. I think I first, the first piece of open source software I ran was while I was in high school. I was pretty sure it was like 1988 or 89. It was a DOS-based fractal generation program. I and then co-founded something in 95 called the Apache Web Server Project became Apache Foundation.
But even like before 95 was using a ton of open source software because frankly, open source software built the internet. And if you were building websites or e setting up email or DNS or anything, yeah, you might be doing it on sun hardware or even so on, on a proprietary operating system like Solaris, but you were probably doing it with send mail and bind.
and, [00:04:00] ftp D and gofer and other things that were open source. And so I was just immersed in that and, while I was at school learning, learning computer science concepts that were 20 years old spending time building things that were brand new. And so I just plunged in, didn't even get my degree, plunged in and worked at a sequence of companies starting with Wired Magazine.
And then another one that built websites for a living. And then I did a couple of startups. I worked at the White House for a few years on open technology under the Obama administration CTO at the World Economic Forum for two years. And then I've been at the Linux Foundation for the last six.
previously to this running a blockchain project called Hyperledger which I've passed the baton on that, and it's growing tremendously now that everyone realizes, , this is not so much about crypto as it is about systems of record and distributed ledgers and the like.
Anyways, long story short my career's been about open technologies and organizing people and organizations to go and have an impact much larger than any of [00:05:00] them could
as individuals.
Eric: Excellent. And maybe a little bit about how you transitioned from running basically the Hyperledger Foundation to the Open-Source Security foundation.
Brian: Yeah,
Both of those who were our projects at the Linux Foundation and the Linux Foundation is a fairly standardized model for pulling together a set of stakeholders.
typically companies, but it can be other types of organizations that are providing resources to run a very lightweight air traffic control function. Myself as gm, I have a very small staff and a modest-sized budget compared to the security teams of even mid-size companies.

And we don't really write the code ourselves or create the content ourselves. Our job is air traffic control, be the support infrastructure for the community of contributors that include the stakeholders, and that by stakeholders companies like google and Microsoft, vendors in the space like Sneak and [00:06:00] Sonotype and Ancor.
End user organizations like banks or energy companies or the but most 'em, critically and importantly, it's contributions from individuals. Some of them working for those stakeholders. Some of them they're simply out of personal interest or are a desire to improve the state of the software security in the open-source world.
Working together across base, a portfolio of different initiatives under one roof, the open ssf and with a very lightweight team managing and facilitating those, the, those operations. But really all the good stuff comes from the community. And that was just like with Hyperledger, the same deal.
Collect a set of stakeholder organizations, pull together a set of software projects, but the real core of it is the open source developers building things that go, that can then be used to go and solve real enterprise blockchain problems. And internal to the LF Linux Foundation.. We just have a really nicely templated system for managing the stakeholders, managing budgets, managing staff, [00:07:00] putting on events.
The LF does north of a hundred in-person events every year that bring together those different communities, bring them together neat physically which is a novel thing now that we've come out of the pandemic. But I really worth doing. And I, yeah, I it makes, it's like this machine that we can crank to, to take on new projects and new domains over, over time as they emerge, like energy or in healthcare or other verticals.
Now that open source is reached up far beyond the operating system and database layers into really all the different domains that software.
Eric: Excellent. Excellent. I was reading the annual report in advance of this call and you're certainly extended into a lot of different areas and hoping to touch upon a number of them today.
But what would you say are the core principles for ensuring open-source software security? Maybe with a little more emphasis on the open source part of it, [00:08:00] as opposed to just standard cybersecurity or I guess licensed or closed systems.
Brian: So I think one, one important thing to keep in mind is that there isn't really much of a distinction anymore on the ground between the world of open source software and the world of proprietary software.
Sonotype state of the software supply chain report suggested that about 80% of the. software stack, whether that's software sitting in your phone or your car, or behind a website 80% of that is preexisting open source code. It's not to say there isn't a proprietary software world.
There certainly is but it incorporates so much open source code inside it that, organizations even struggle to identify how much open source they're using, right? We

saw this with the log for J log for Shell vulnerability thing that's led to calls for software, bill of materials and the like.
Is something helping under organizations understand how much of that they use is really key. [00:09:00] And no one can guarantee that there are no, no software defects or no, no security holes, right? Software is complicated. No amount of machine learning we're gonna throw at this stuff is ever gonna make it defect free.
I and show me a hundred lines of code and I will assume that there are a couple of security holes latent in that code, no matter how many people have previously looked at it. . So we have to do instead is start talking about risk and start talking about what's the risk that the given body of code or a collection of massive amounts of code have vulnerabilities that are yet to be discovered or even vulnerabilities that we know about but aren't, just, aren't as good as we could be at knowing that and managing that risk when it comes to updating and pushing out new versions and keeping our end users of that software from having their data inadvertently, right?
So open, the Open source security foundation is all about trying to understand [00:10:00] how to quantify that risk for giving piece of software or software that's used. How to address certain key weaknesses in the way that software is built and deployed. Call it the software supply chain. Totally thrilled about that title, about that concept.
But we can talk more about that in a bit. But that there've been opportunities that for people to inject certain kinds of attacks that we just didn't think about 20 years ago in a, at a time when the open source world was a lot more high trust, where generally speaking, people pulled tar balls of code down and compiled it and installed as route without blinking an eye on your personal systems.
So Open ssf is about tools to measure risk, tools to mitigate these sources of risk in the software supply chain. . And then also educating software developers about ways to avoid common pitfalls when it comes to security. Most of the CBEs that are out there actually use a very [00:11:00] small set of approaches to compromising.
And you can actually, teach people about here are the 10 or 20 most common tropes that, that mistakes call them, that software developers tend to make that can be exploited. We actually have courseware around this to teach that, which include things like, don't trust user contributed input.
And if you don't parse it for format strings, which was one of the core root bugs in the Log4Shell vulnerabilities. So those three things are the rough frame within each of those. We've got a dozen or so, or more active projects that manifest each of those three. So happy to go into more depths.
Eric: Absolutely. So may I don't know whether we should talk a little bit at this point about the mobilization plan and its goals or whether you want to get into one of those three buckets that you identified there. I guess basically the mobilization plan encompasses all of those, but [00:12:00]
Brian: It, it does.

So the mobilization plan was devised really in the aftermath of Log4Shell. Log4Shell caused a whole lot of open source projects to get a whole lot of renewed attention and focus and questions. I think arose that hadn't arisen before about is this software, is open source software in general, software that can really be dependent upon for critical infrastructure, for banking, transaction networks, for energy grids for these kinds of things when something that seems like, and always retro hindsight is 2020.
But seems like a simple bug, quote unquote, and yet I, it seems so hard to discover, and remediate and get those fixes pushed out and it begged The question was, this is, is there too much risk involved in not just open source software, but in the digital infrastructure that we've built so far, right?
Have we gone too far into the move fast and break things [00:13:00] landscape, for things that people are depending upon to, to, self-drive their car, for example. The reality is that open source software has always had both a great reputation for security and some real issues that have needed addressing that people who've operated here have known about. And yet being able to organize enough effort to go out and tackle them is, has part of, partly been the missing piece. And this isn't because it's trillions of dollars or billions of dollars of work, but because there's certain things like auditing source code, paying a third party to go and scan, look for, bugs, that could be bad bugs or the I, to do that at scale does require some real resources and we've been able to pull together some, but it seemed to be a moment when the time is needed to articulate a set of additional things that could go off and be done.
If we're starting to realize the cost is in billions of dollars in terms of lost productivity or data disclosures or fines or other [00:14:00] things. Really cost costly to society. Where might, if we can say an ounce of prevention is worth a pound of cure, where are those ounces of prevention that we might go and find?
So the report, the mobilization plan was devised in that spirit. We worked after a query from the National Security Council Anne Newberger there who had cybersecurity for the N C at the White House said I, it is, what's your plan for actually getting these interesting ideas you have implemented so that, we can, we won't entirely avoid the next log for J but so that the risk of the next, log for shell log for JB breach is mitigated and we have a better time of responding to it when it happens.
So we put that challenge out to our community. We organized working groups around a couple of thematic areas. We said, money, it's not that money is no object, but I, if you had enough money to make. double digit percent impact on a problem space that you've identified. What would you how [00:15:00] would pursue that for the first two years?
Just a, think of them as startup business plans essentially. And we said let's look at, what it costs to do each of these 10 and put them together into one, one package and thinking that might be billions of dollars. We were surprised when we found that you could actually have a tremendous amount of impact with some minimum viable approaches for actually 150 million over a few years.

A number that we then did some fundraising for our own community around. We've started to get to work on some of those different items. And I won't go through the full plan because, time is limited here and people can read it on the website. But they call for everything from implementing better scanning of open source projects for new vulnerabilities which we've implemented through a project called Alpha Omega.
I can talk about it in a bit to launching an incident response team. Basically a, a band of superheroes who you can call, put up the bad signal if [00:16:00] you're an under-resourced project and you discovered you have a bad vulnerability and you want to figure out the right way to get it fixed.
And coordinate the disclosure process around to simply figuring out how do we take these educational materials we've developed and scale them out to a point where everybody who calls themselves an engineer has taken the course, or understands if they're writing open source code. Here's the checklist.
Or their consuming code all these different kind of attributes to it, which again, map to ma understanding the risk being able to mitigate that risk and raising the floor on, on how well educated we all are about how to write secure code. So that's the mobilization plan and as a big picture and kind of a, as a true north document for us, as a community.
Eric: And one of the things that resonated was the notion that patch patches in open source aren't pushed like a proprietary. If you have Microsoft, you get your [00:17:00] updates or you have windows, you get your you have updates or whatever proprietary system you have, you anticipate that there's going to be updates to the system that resolve security issues and flaws and patches, but that doesn't exist in OSSF, in, in open source community or to what extent does it.
Brian: So, there's an
important thing here that has allowed the open source community to scale really well and be a safe place for people to collaborate and for companies to open up a lot of code, which is that it's tenant to operate on the assumption that the end users are well-informed, are able to get smarter about a thing they care about, are able to see the code and how it works and see the community and how it builds and that they're able to bear the responsibility of updating the code.
And, if there are problems with that code, that kind of the responsibility is born by the end user because they didn't pay for it, right? If I have an apple tree growing in my front yard and you pull an apple [00:18:00] off of it and eat it, which I'm happy to let you do cuz it's a but then you get sick, right?
Am I liable for that or not? It's not. I'm not selling you an apple. I'm giving you an apple. Now that has created a tremendous economic opportunity for companies like, red Hat and Canonical. Basically, every Linux distributor every software re distributor to be able to incorporate open source into their products and services and push it out even if all they're selling is a support contract.

And right now I'm running, talking to you, I'm running a BOUN two, every morning I get the kind of, here's the today's updates for security and functionality, right? Then I go ahead and just scan it and say, yes, generally I should actually just even push it to be automated at this point.
And what it's doing is it's updating open source code, but it's open source code that canonical in this case has said here's the updates that they have confidence in that they guarantee as part of their distribution, right? So that model works pretty well. But it does mean that [00:19:00] companies that pull and dev individual developers who pull open source direct off of GitHub or direct off of a distribution point Pipi or N P M or Maven or the bear a fair degree of the responsibility of making sure that code is fit for purpose, right?
And that's where there's been a gap because what tools do developers have to understand? If I'm looking for a Java logging framework, like Log for J first off, I might just choose the one everyone else has, which, has been logged for J right, which probably is not irrational, but doesn't create the opportunity for others to come in with different priorities, such as a priority that might emphasize security over feature.
More often people look for a function. They go type it on GitHub or do a Google search or something, and it, and they'll tend to gravitate towards the ones with the most stars on GitHub or the most sense of ac an active community, rather than necessarily the one that has been built with a set of practices and principles that tends to lead [00:20:00] to more secure software.
For example, when is the last time there was a third-party audit around this code? Do we make sure that is this code maintained by a group of people or one overworked volunteer and they're trying to make it all come together, right? All these sorts of attributes that I can go to. Details on how we, the open ssf provide tools to try to discern the more securely built software from the ones that are a little more haphazard.
Let's just be brief, Frank. There's a lot of haphazard open-source code. And with those tools, developers can steer towards more secure alternatives and organizations can try to figure out where's the code of the thousands of packages I'm using? Where's the code that's most likely to be a problem, and should I make a decision about that?
Or maybe I should even invest into that, right? You could see one of the reasons why Log four j didn't have the community around it that it deserved looking for some of these security holes is everyone was able to take it for granted. [00:21:00] They just assumed it would be secure enough, partly because it had Apache's name on it, partly because it was so widely used elsewhere, but they took it for granted.
And tools to measure risk or also tools to create a roadmap for investment into improving.
.
Eric: And so you were talking about some of the techniques that you could use to I don't know, quickly, but to assess whether some core security controls whether it's been audited, whether it's supported by a larger team commitment level.

I guess ongoing scanning might be another one. How, is there an easy way to identify that or is there a red flag that it's not being done?

Brian: So there's a bunch of human tools say human tools, human level tools to that, that do require a little bit of work by the developers to deploy, but can really help signal whether this code has been built the right way or not.

One of those is [00:22:00] we have a set of best practices for people who are using or producing open source code as. , those who are consuming that, and they're basically, they're called the concise guides to, to securing open source. They're available on the Open SSF website. We have the course that I mentioned, developing secure software, which is about 16 hours of courseware.

You can take an exam at the end of that and get a certificate of completion. That shows that, you've actually been through the content. You can answer questions about it and use that to prove to put it on your LinkedIn page, whatever. I'd like to be able to see in the open source software I use how many of the core developers have taken that course recently, right?

There's another human tool we have called the security badge. There's a little bit of that's automated, basically invalidating people's answers, but it asks questions like, do you have a security team? And do you have a security process around accepting a pool request?

Do they require sign off by multiple [00:23:00] people? That sort of thing. And then we have an automated tool. we call the security scorecards, which looks at GitHub repos and it's being expanded to other repos kinds of repos as well. But it can look at a GitHub repo and ask it questions that are heuristically driven.

Things like, do you have test scripts? Everybody that does of, sure, but do you use fuzzing in your test scripts? Now that might not be appropriate for some languages or some projects, but generally speaking, if a project has fuzz tools brought into their test scripts, that's a sign a signal that they probably take security more seriously than the average project.

So take a hundred of these heuristic tests that you can automate, apply it and look at these repositories and develop a score. And that's what scorecards does. Score between zero and zero and nine. That will tell you, think of it like your credit risk, right? What is the risk that there's an undiscovered vulnerability sitting in this code?

We also, there's obviously tools out there from vendors that do kind of dependency analysis that point to if you're. [00:24:00] pro, the thing that you're downloading from an open-source project contains dependencies that have updates that have addressed known security vulnerabilities. There are scan tools to look for that kind of data as well.

So, pulling all that together into one kind of dashboard is a project that's underway, but we've got a lot of the pieces together, both for automated tooling and for human tooling to be able to help assess that risk and steer investment and steer developers towards making.

more secure decisions.

Eric: So a lot of projects that use open source as a prerequisite to going live, like particularly in the digital asset ecosystem they'll need to have a code audit. But from a user who's u who's leveraging it how can they distinguish between. , a good code audit, like if they're not actually reading it.
Cause everybody has, in in digital assets, crypto, it's, always TLDR too long didn't read right. Yeah. How [00:25:00] can they quickly differentiate between a project that has gotten like a very thorough security open source security audit versus one that maybe cut some corners?
Brian: Yep.
No, that's a great question. This is a domain that probably benefit from some standardization or even certification. In the, for the time being, there's this great organization called ostif, the Open-Source Technology and Improvement Fund, which is a, an organization that acts as a front end to a pool of security researchers and firms that they've vetted, that they work with to perform this kind of work for open-source projects. So I can go to them and say, I want to get an audit for. , the upcoming zero, something got zero release of a package. They can farm that out to this pool. The pool comes back with quotes. Generally speaking, something that's, moderate complexity, a couple hundred thousand lines of code will be about $50,000.
[00:26:00] Obviously Austin also makes sure that the open source project is willing and interested to work with the auditors because this is not something that can be done without the cooperation of quarter maintainers. To understand or to help explain the internals of code, help the auditors distinguish between something that looks like a bug and is actually a feature.
Sometimes that happens. And then generally the auditors come up with a report and the basis of that report can serve for fixing the bugs that they found. Or thinking even deeper. Sometimes an auditor will say, this feature that is really. And this is part of the issue with Log for Shell.
This feature that is really complicated does, is really powerful. But it's like putting a chainsaw in the middle of a dinner, cutlery set, knife, your fork, your spoon, and your chainsaw here. And somebody might cut their fingers on that chainsaw if they don't know how to use it, right?
Or, it might be a source of attack. Or of compromise. So, I so that's what a good, a really good security audit can do. And [00:27:00] as I mentioned, ostif has been good at setting a pretty high bar there. But we need to scale that out because what we'd like to see are not just so the Linux Foundation, our communities are able to fund audits from time to time. I, I, we've also at Open ssf been working, doing grants to organizations like Python and Go and Rust, I'm sorry, Python and Rust. MPM and a couple others to do their own audits and do some other security work. But by and large, most open source code is not third party auditing. What if we could take.

I wouldn't be able to tell you who the next log for J is, but I could probably say, here's a list of the 100 projects that have the greatest likelihood of being the next kind of source of a major compromise based on their criticality, based on low scores on these other tests. Right? And so why don't we get th do go find a funding source 50,000 or even a hundred thousand dollars times a hundred projects to go and audit the [00:28:00] top 100 open source projects as mentioned. That would cost 10 million a year, maybe 200 projects, 20 million a year. Think about the amount of risk that we could drain out of the system by doing just that.

So, it's that kind of coordinated large scale effort that we'd like to do more of. That is gonna call for a lot more firms are able to do. And people have to trust that the audits actually generate. Are a quality scan that can be dependent upon by third parties. So back to your original question it would help to have some degree of certification or some degree of quality assessment here right now, though, there are plenty of firms who can take this work on and do it a great job.

So, we just need to use that more.

Eric: And I guess it's, if you have a community that supports an open source project, they can continue to make improvements over time. But to the extent that open source means also not independently funded, then th this question of how you continue to ensure that you're getting the [00:29:00] requisite audits over time becomes critical.

Brian:

There's not an, generally speaking, a sustainability issue for open source. Most projects are able to do quite well by having, harnessing the volunteer labor. of developers who either personally have a stakeholder interest in the outcome of the project or work for somebody who does and is able to get paid by that stakeholder for their work.

And I you do hear a lot of projects where it's one developer, they're struggling because a hundred thousand people are using it. When you hear that, you should also go why aren't there other developers involved in that project? Why aren't some of those a hundred thousand able to become co, co maintainers?

And generally, there's usually a kind of a community management issue at the heart of that, that that needs dressing. But the by and large the most successful projects have teams of people, and yet they're still volunteers. Typically. Sometimes they're able [00:30:00] to put together funds like we do with the open ssf through stakeholders for different projects.

But Log 4Shell, for example, is really just driven by that time that they were able to carve out from their schedules to work on features and to pay off technical debt and to obviously fix bugs, but paying a third party a hundred thousand dollars for a third party audit, the core developers just didn't have that in their back pocket, nor did they have the, a point in time where they said we should really pool our funds or go get some grant funding somewhere to go pay for that to be done.

So, I think there's little un these edge cases where there's a bit of a market failure in open source code, but they're not, it's not in insurmountable. It just requires us to have enough kind of collective awareness of the problem, of the need for it and set up some additional funding structures to be able to go and do that kind of work.
Secondarily there's other investments that are kind of p. In nature around security that don't create immediate [00:31:00] benefit for you, that are that for developers and so they tend not to do things like implementing multifactor authentication on their repositories or requiring that to be a contributor to a project MFA is a little bit of a burden getting to publishing a software bill of materials for your project right now, still a little bit of a burden. If we can make SBO m generation basically zero cost or other kinds of security practices as cheap as possible, then we'll see more people adopt them. But we still have to recognize a lot of this stuff does require investment up front. We need figure out how to incentivize that.
Eric: How does that interact or how does that impact the signing of code?
Could you explain like how signing and attestations and grading can all work to better secure open source?
Brian: Yeah. So in the good old days, people would post, gzp tarballs of source code on random websites. You'd pull it [00:32:00] down, you'd build everything locally it and you'd run make install blindly knowing that, hey, if somebody was smart enough to know how to build a tarball of source code and then compiled, yeah, feel free, go and install it as root, because I'm sure somebody would catch it if it did something bad.
We can't. All those assumptions we made back then, which were useful to bootstrap with, and it kept things simple. We just can't afford those assumptions anymore. And so, the first part of that is that, I, developers when they're working on, really important source code, you really care about making sure that it's hard to steal their credentials, right?
So that's why multifactor off whether. A text message confirmation or a hardware fob or, an authenticator app on your phone. All of those are things that make it harder to steal someone's credentials and flip in and update into some core critical piece that's pulled in a million times a day into people's production systems that they just assume is gonna be safe.
We, [00:33:00] that's like a basic level of protection we can do, right? Is to make sure that people, when they're dealing with developers and development teams that those teams are stable and it's much harder to compromise those teams. The second is no one builds locally anymore. You probably do at the tail end of a development process, but we've been so trained to pull binaries down and assemble these binaries into system images and do that, frequently.
That this question of what connects the source code for a module to the binary that I've pulled down is a big question. And so that's where a project called SIG Store comes in. six store is a way to attach signatures to these binary objects as they flow through. Again,

apologies for the terms, the software supply chain so that there's a tighter bond between you as the developer and the developers of the modules that you're pulling down.
Much the same way that, before TLS on the internet, you could go to a website, you could see the, in [00:34:00] the URL, the name of the website, you attached a lot of significance to the domain name. But you really had no other guarantee that you were talking to the people you thought you were talking to, and that your communications were secure between you and your bank or you and amazon.com.
And so TLS came around and we secured the web. And actually I like to make the point we didn't secure it overnight. It took 15 years to go from a web that was predominantly unencrypted to one that was actually, I think we're added 95% now. and it happened through a series of stages that started with incentives.
Carrots as I call it. Like you want to see that. Cause you want the little green thing on the URL, on the web browser. So, you as a website, you wanted to, you'd go through the extra effort to get that TLS certificate and the complexity of configuring it and making all the pieces work well, and then updating it once a year.
And then we got to a second phase, which was [00:35:00] about shifting the defaults. And in that phase was when you saw lets encrypt come around, which made getting a certificate and renewing it every 90 days, much more efficient. And that allowed almost every new website set up from that point out of the box supports security rather than it being something that people had to remember to do and add afterward and have some complexity around it.
And then the third phase is the sticks phase. The that phase where websites today are significantly penalized if they don't use tls, right? It'll show up as a security warning in a web browser. . So most of these security practices are gonna need to go through a phase of, carrots and then defaults, and then sticks.
And so one thing we're trying to figure out now is how do we make more of the things that are promising carrots where the people who are really on the leading edge of this stuff have started to adopt it like six store. How do we move that to defaults? And so, with Sig store in particular, and Si store by the way, takes a very let's encrypt approach to managing keys.
They're very [00:36:00] short-lived keys. They're keys that you can automatically grab as part of a bill process, but the signing is still intended to be done by a human. And so that you have that close connection between the humans working on this code, and the binary that I'm pulling down and incorporating into my build.
And so now the stage with sync store is it's hit 1.0, general availability, the cloud native community. Massively moved to it. You can now get cloud images that are signed one or two levels even back, I think. So, you have high assurance that the code that you're running it works for that community.
Now we need to figure out how do we get six story embedded into the tool chains in of the world, in places. So it's as a default in your, pulling packages off of pi from Python, for

example, or rush crates or those kinds of things. Eventually, our hope is that not only do all the signs, but that signing them is a part of the process of getting things into those repositories.
So anyways, long story short that's the story around [00:37:00] signatures. There's still more to answer though. It's not enough to simply look at a signature and it checks out green and it's done. I talked about building locally, with source code understanding how code is built, who has access to the build systems, what are the policies around building it has driven the need for a couple of different standards in this domain that build on each other. One of the cores of those is a standard that we have at the open SSF called SALSA for security levels for software attestation. And it describes a set of levels one, two, and three that speak to increasing levels of security around your build and distribution process. And so, if you're a bank, or you're some other high security domain, you could require that all of the components follow salsa level three.
That's something the healthcare domain is already actively pursuing this kind of thing for medical devices, for example. And that might include things like publishing SBOs and the like. But it's a way to try to characterize [00:38:00] the processes used in a computable way so that at the end of that you can say verifiably have got a, an image that has been built to the standards that I require for security.
Eric: Excellent. And then grading is part of that as well, right?
Brian: Grading what?
Eric: Yeah. The notion of, I, I was going through your open your annual report and it said it referred to the sign signing, attestations and grading.
Brian: That's the one, two, and three.
Eric: Okay. Got it. Excellent. And so when you consider the advances of the threat actors in this space, the advanced tech technological capabilities, what do you see as some of the, forward-looking risks that, O S F is going to have to deal with and developers are gonna [00:39:00] have to start thinking about more carefully?
Over the next few years,
Brian: I'd say I, I'm worried less about machine learning and ai, and I'll get to that in a bit, but more worried about the fact that so many of the vulnerabilities that we find out there are due to things that we know about due to bugs patterns of bugs that have lived with us for a time in memorium around unsafe memory handling, unsafe handling of user contributed input the kinds of things that you can systematically scan for in some cases. And identify and perhaps even try to remediate at scale. So this is a big thing we're aiming to do with the Alpha Omega project, which is to build on the work of a researcher we've hired named Jonathan Lecha, who has I, had run a couple of different campaigns. We've identified a bug that is common to a large number of open source projects.
For example the insecure [00:40:00] use of the zip library in Python, where if you run it as route and you expand it and you let it override your, slash directory, obviously it could overwrite ets see password or anything that it wanted, right? Now securing that use was

something that the developers of the module set is up to the application that depends it, right?
That includes it in and you can run a scan that tells you which of those applications actually sanitize the environment correctly and which ones don't. And you, when you run these scans, you find thousands of open source projects that don't do this right. and you can tell 'em about it.
You can even submit them a poll request, which is work that we aim to do, so that it's easy for them to adopt a fix for that. And yet it's still dependent upon the developers to see the need to implement that fix and actually accept it and push an update out. And so I worry a bit about not the leading edge of this, but the trailing edge.
Like how do we really get more of for the, at [00:41:00] least the software that matters, we're not gonna be able to secure everybody's, random school project or weekend hack that's on GitHub, but at least of the stuff that's included by the thousands or millions inside of build systems of the world.
Let's try to look for the easy stuff and get that remediated and monitor for anybody who inadvertently puts something easy in. And then the second is there's some harder stuff to do. I there's implementing the kinds of processes and the tooling that we talked about that would try to catch attacks on these systems or making sure when there's a new vulnerability announced, there's.
A bug in one module is likely to be a bug in others as well. And there's still way too many systems that chip with dependencies. So that same sonotype report noted that was 75% of software today that you can pull down from an open source repository, still ships, including dependencies that are more than four years old, and for which there's outstanding vulnerabilities.
So [00:42:00] this is still somewhat of a crisis. We're still shipping lots of insecure code. And we just need to find ways to, to modernize so much more of what we're shipping and do away with that close that window, make that window smaller. Finally, back to your actual question, the leading edge of this stuff, we still depend, even if we do all the things that we've talked.
we're still gonna depend quite a bit on the integrity of developers themselves as a proxy for trust and code, right? People use the Linux kernel partly because they trust Linas and they trust the development team around it, and they trust the processes. And the Linux project has been successful because it's been able to scale out and have this massive number of people looking at it.
But I still will argue, the day leans retires is, there's gonna be, somebody will have to step into his place to take on that mantle of community trust, right? And community trust is important, not just in like the project leader domain. But if I wanna [00:43:00] start contributing to an open-source project, I have to earn my stripes by offering some bug fixes, offering some documentation improvements.

I gotta build that reputation that can be gamed using machine learning and ai to the point where I would worry about being able to automate. , the process of going and having conversational AI is going open a lot of new bugs some of which might actually be issues some of which might be simply a way to create noise.
And we know that spamming of a community is, can be really deadly to that community. And it can cause them to miss something. We might, bad actors might even be able to use machine learning models to figure out how to slip back doors that are harder and harder to notice, right?
It's almost if you develop a machine learning model to note it, to look for back doors in contributions, you're also gonna be training a tool that can be used to develop better back doors., it's kind the two, two sides of [00:44:00] the way that we do machine learning. And so, I that worries me a bit.
I'm also worried a bit about copilot in so far as it recommends and helps you write code if it's trained on source code that has bugs. , will it be introducing bugs in its recommended auto completes as developers are typing? So all these are currently like future domain things, which we are, we, there's a bunch of conversation about way that, ways that we might try to be resilient against that.
But fundamentally, social hacks are social hacks. It's, there's no tooling that's gonna prevent or eliminate kind of exposure to some of those risks. We will, in the meantime, there's a whole lot of low hanging fruit that we're gonna be, I think more focused on just cuz there's so much more of it more work to do there and so much risk that we can drain out of the system.
And we will watch for, as people develop techniques to mitigate the leading edge of that. But I, yeah, it's important to, to monitor,
Eric: which I [00:45:00] guess, all that kind of goes back to signing and at and attestations and ensuring that there is reputation of the developers who are contributing.
It seems like that's gonna become. just increasingly important, because in the absence of that, you could have AI arguably developing that reputation, right? Over time. To your point, you could open a lot of bugs and you could assign reputation to it. Maybe reputation doesn't account for the bugs that stay open, right?
You open a bug, it never closes. Oh, high reputation, he opened a lot of bugs, but, did they go anywhere? Were they real bugs? Were they basically designed to overwhelm the system, the community?
Brian: And it's, it is true. We do depend upon people to and reputations to really provide some of the safety and guarantees.
I'm very much a trust but verify kind of person. It's the only thing I'll [00:46:00] pull forward from the Reagan era. And better tooling to help us verify is gonna allow us to more easily trust., we have this special thing as well. I think in the open source world where it's still the case that people can participate, not anonymously.

Anonymous is a very loaded word, but su anonymously, the easy example is Satoshi Nakamoto, . But there are other examples of people out there who have been participated in core projects and either never revealed who they actually are or selectively revealed that. And it's a good thing that we're able to allow that kind of participation because frankly, standing up in front of the world and creating a creative word is also an invitation to a lot of, not just criticism, but hate male if you're the wrong color or the wrong gender really other unfortunate kind of circumstances.

People need to be able to participate synonymously and open source. And we really are best when we're tapping into the collective wisdom and skills of people around the world. [00:47:00] And, there's a lot of rise of nationalism out there in all sorts of fronts. And I, so far open source has been a sanctuary, a refuge against some of those pressures.

25% of the projects are created last year in the cloud Native Compute Foundation were created by developers as best they know, as best they can tell developers based in China. There's a lot of global collaboration around core pieces of open-source infrastructure. And we have to be able to preserve that.

And I think we get there partly by basing on our reputation but also by using tools that allow for verification in all sorts of forms and help us catch back doors and help us address the low hanging fruit, which is always gonna be more widely deployed than the advanced world. Yeah.

Eric: You didn't choose an easy area.

Did, I guess at the beginning of last year, there was a White House they arranged like a summit [00:48:00] that to focus on open source security you, your group to followed up on that summit to try to build on that. O other than the obvious, what do you think pushed the White House to do that then?

Other than the obvious? Like what, like I, there solar winds for example, was a big concern. I don't know whether it directly fed into, an open-source initiative, but what do you think were the events that specifically drove that?

Brian: So yeah, and SolarWinds wasn't so much a an open source thing, but it was certainly a supply chain. Correct. It's certainly the case where, it got itself into this critical piece of widely used software that ran with elevated privileges most places it ran. And that was a wakeup call from a supply chain point of view.

And then Log for Shell was a wakeup call, partly from a, it was difficult to figure out who was running it and how exposed they were. It was [00:49:00] really easy to compromise. I and I, it was an example of a lot of things that were broken in small ways that in combination and.

demonstrating brokenness in big ways. And I do think the White House was reached out as they do in other circumstances. They did after Harley. They I, it depends on who's in power, but generally speaking I see the US taking a tone of appreciative inquiry around things that, that they don't understand, which is perhaps unusual.

But I, in this case in particular, they wanted to figure out not just how can they help, but how is industry actually rallying to, to address this issue and what weight can they bring to bear? And also how do they avoid coming up with regulations that might inadvertently address might address the problem, but inadvertently caused a lot of other costs to be born.
And turned down a lot of innovation or a lot of the [00:50:00] pace at which open source works that would have detrimental impact, in other ways. And I can I contrast this actually with. An approach being taken in the European Union around some legislation that they are proposing and now working through their member parliaments around, called the Cyber Resiliency Act, which also is concerned about security, but which pushes it very much onto the producers of code, whether there's a contractual relationship or not, and says, Hey, if you're an open source developer and you put something up on GitHub and other people use it, you're basically, it's like giving toasters away.
And if you give away a toaster that catches on fire, you're still liable for it. And that's a problem when you think about, you can't necessarily control, first off, there's no such thing as bug-free code, right? And every, any suitably humble software developer is gonna realize there's gonna be code, bugs in this, that the wrong person using it in the wrong way, is going to inadvertently [00:51:00] have a major issue with, and now I'm gonna be liable for that.
Eric: Yeah. And I believe that there's a products liability act as well in the eu that also raises some question directives. A directive. Yeah, it's a directive, right?
Brian: So
it's well intentioned in that everybody wants to see certain practices picked up, fewer known vulnerabilities and shipping code.
I brought that up too, right? It'd be nice to close that window. Every, everybody wants to see more software bill of materials. We're not quite at the point where it's zero lift for developers to, to create those and consume those in meaningful ways. Everybody wants to see better processes adopted, but it's like I talked about with carrots than defaults, than sticks.
So much of this is even before the carrots phase, let alone in carrots, and we haven't quite shifted to the default. But more importantly, shifting that liability to people publishing open-source code even if there's not a contractual commercial relationship. Makes it extremely high risk for people to release open source code.
And when you do that, you take away a lot [00:52:00] of their incentive. It's not, they don't have enough other incentives to counteract and you're gonna see people fix bugs, but keep those bug fixes to themselves cuz they don't want the liability of fixing a bug. And then other people, depending upon their code, why would you share it?
So that is a major concern of mine and it's something, or late last week, the White House did issue an update to its cyber security strategy. It does call for liabilities to be pushed onto the companies that are selling software and providing digital services. It seems to be

a much clearer scope of where that liability rests, which is between a vendor and a customer, rather than on anybody releasing open source code.
And that's, and I think that's the appropriate place if you want. Liability protection. If you want a warranty, go and buy it. Go and pay for it. And companies will sell that to you. I, and I, that's the appropriate place to, to apply that lever [00:53:00] rather than applying it upon. Somebody who finds a bad security hole, fixes it, puts that up on a website somewhere. So, it could be part of the fix. And yet it has a hole , which often happens. Often. The first fix for a security hole doesn't quite fix it entirely. And creates a second hole somebody can quickly find out. And you don't want then that to mean people don't share security fixes anymore. That's we are finding ourselves being pulled into these conversations quite a bit. And it's not what I'd prefer to be doing, but it's important to do because it's important that as the world understands how to consume this software in a safe way and how to support it as well that we're involving the governments and in that, and regulatory bodies and other kinds of stakeholders in that process.
Eric: how does I guess your focus, what you gauge as the focus of others change with regards to permissionless code, like truly permissionless [00:54:00] systems? Because in permission systems, you arguably have, there's more centrality, right? There's ability to coordinate maybe potentially differently, right?
Permissionless, you can still coordinate, but it's a, it could be a more dispersed community.
Brian: So you're talking about this a little bit more
in the context of blockchain technology and distributed ledgers and the difference between permission ledgers and permissionless ones like Ethereum, Bitcoin, that kinda right.
Eric: Or just even more centralized versus decentralized systems.
Brian: Well, I, I, so when I led Hyperledger this, there's a question that came up pretty frequently because we were explicitly in our Hyperledger explicitly is blockchain for enterprises. And enterprises when they work together to build these networks for sharing information for coordinating digital processes, they have a governance model.
And that governance model can be something that's implemented by [00:55:00] neutral third party that they all invest in. Like the Linux Foundation, the way that we're a governance body for the development of code. Or it's through a set of standards or through an, a regulator or international body of some sort.
But you never have a business context that doesn't have some governance function in it that the stakeholders are funding or are otherwise a part of. And so, the underlying digital architecture to support that it seemed to be permissioned distributed ledgers, blockchains, or the nodes are registered with a central actor.
They get a either a full copy of the data set or some agreed upon subset of it or something and they're able to operate and function. And actually it saves a whole lot of, obviously, energy consumption. You don't have to do proof of work. You can need. Consensus

mechanisms that are just very different, proof of stake or simpler ones around forming consensus on a ledger.
So, all sorts of operational benefits. But it also mapped to how the business world works when that's done wrong. The [00:56:00] governance function is a, is too much of a limiter on collaboration, too much of a limiter on innovation, and it develops a bad reputation. And there's an awful lot of examples of it done wrong.
So it's tempting to think that it's, not possible to do it right. And yet I've really never seen a collection of business processes outside of the black market where permissionless transactions are the standard, right? Are the way things are done. Bringing that back to technology, it
Eric: really is.
Would you would you put Ethereum in the bucket of a permissionless network?,
Brian: I think proof of stake is permissionless. And I think most of the interesting things implemented on top of Ethereum have governance functions in them. Every Dow is implementing a governance function. And there's always an implicit governance in the form of, upgradable contracts.
Who are the developers able to push those upgrades, right? But that gets papered over by this community. So, governance, [00:57:00] right? There is a spectrum. Very rarely are you all the way at the anarchist point on that spectrum. Usually you're, somewhere in the middle if you want to balance between being efficient and being flexible and facilitating innovation.
While still ensuring bad actors get rooted out. Bad abuses are contained that data that should be transparent is transparent and it's generally useful to try to think about what's the minimum amount of centralization you need? to be able to get the maximum amount of benefit, right?
I do subscribe to the power corrupts kind of philosophy, right? So, let's, how do we create the smallest power structure necessary to make the whole system work? And outside of the blockchain space, I think the best example we have of that is I can, right? Who have been able to successfully warts and all, totally not beyond criticism, but by and large the internet domain name system has been up and able to function.
And we've been able to build [00:58:00] great things on it basically by having I C A N focus on one core problem and build a really carefully engineered set of contracts with domain name registrars to ensure that your domain names are portable from one to the next, right? And they've had to stand up to sometimes more successfully than others, but to governments of the world that want to take away domain names because they don't like what you're doing on your website, right?
They have done that to certain requests. There were about clearly illegal activities, right? Perhaps more than I would've liked, but that's allowed us to ensure that the internet in Saudi Arabia is that domain names are just as resolvable in Saudi Arabia as they are in the

United States, right? So let me just submit that models that look like I C a N and other domains are interesting.
Let me bring this back to security. Every place that we can say security is on the shoulders of those who are deploying these systems to appropriately vet code. The generally the better, the more [00:59:00] scalable we're gonna be. And yet it's unrealistic to ask people to scrutinize every line of code that they're running inside an application that they're deploying.
We're building on the shoulders of giants every time we write a mobile app or build a website. And we should be the question is how do we get confidence in these underlying pieces? How do we beyond just a claim by a developer? Just trust me. I put the right bits together for you. How might we use a combination of digital signatures and SBOs and salsa and other things to have an enough of a risk balanced picture to know here's what I can do to lower my risk.
If it's not low enough yet, how do I calibrate it with other risks that I take in the world and what can I do to spend money to reduce my risk? Because then you, then it becomes an actuarial function, then it's something I can buy insurance against. And that's a way that we get to a better, safer world, I think, than the alternative.
And that will require some lightweight coordinating entities to provide some common services. Even the signing service around SIG store [01:00:00] is run as a distributed ledger. It's not a permissionless ledger, but it is one that anybody can read, anybody can write to. And it is suitably distributed around the world.
Eric: Excellent. And going back to the alpha Omega project, which we were talking about earlier. I think we touched on the funding, but what is the construct for both the funding and the determining the recipients? Is it truly being it by need, high level use and need? Or can somebody say, I'm a disaster, I need help, like what's the,
Brian: So let me start with a bit of background.
Alpha Omega was as a project, came from a white paper written by Michael Sveta researcher, who's been involved in this stuff deeply for a while, saying the two things he wanted to put together under one roof, kinda like a peanut butter and jelly sandwich is [01:01:00] scanning of large numbers of open source repositories for common bugs that can be identified and remediated at.
right? And again, this work I was describing previously that's one half of it. That's the Omega side. Go and go for that long tail of open source projects that are still critical, that everyone's using. Maybe 10,000 is the right number, maybe 50,000, maybe that's a variable. But pick some number at the top end.
And go and scan them regularly for low hanging fruit when it comes to security vulnerabilities and get them fixed. And that's infrastructure that has been built that is running today. We've already identified a couple of CBEs and our hope is to get to the point where we're identifying several a week through the use of the system.

And it's not just identifying, it's also reporting it upstream and working with the upstream developers, getting their attention,, getting their trust but then also coordinating a vulnerability disclosure process with them. Because a haphazard disclosure process is like what we had with Log 4Shell and I created a [01:02:00] lot of panic for people over their winter holidays last year.
The other half is alpha and the other half is about. Looking at the largest open source projects and really the lar the larger foundations around those projects, but sometimes individual projects and asking, do they really have the capacity to meet security issues and manage the, their risk for their end users.
When that isn't something that most of these foundations or projects are started with the idea that they would do, right? They've had this a little bit foisted on them and most of 'em have had a security team, an email alias that goes to people confidentially to report it whole.
But very few of them tend to have, staffed security experts on hand to help respond quickly when something happens. And beyond the resources of volunteers, very few of them have implemented the kinds of practices or the kinds of tools that would lead to more secure code. So Alpha, what Alpha has done with five [01:03:00] projects, Is go and make about $400,000 a year grants some larger, in some cases to, to those organizations to help.
In a, with a very broad set of here's things you could go do but help them build a capacity inside their organizations for taking security issues more seriously. And in some cases, that's paying for a third party audit of code in some cases that's paying for a human to, to be the security maven for them, to answer security issues and the new, other proactive work.
And some of those things are about adoption of some technologies open ssf or not. But the idea is that we would come back in a year and do a renewal of that grant. , but over a couple years tend to tail off and demonstrate the value of that to the community itself so that stakeholders in that community rally their own resources and start funding that in perpetuity.
So this is like a bootstrap, it's like training wheels for open for foundations. And they've been [01:04:00] very appreciative of that work. Rust, Python, mpm, sorry. No js and the Eclipse Foundation, and we've also done a bit of a grant to jQuery because something 80% of the websites out there are still running jQuery.
And most of those on old vulnerable version, old vulnerable versions that, just being able to get those migrated and in away is going to have a huge positive impact on security on the web. Long story short, alpha Omega's kind of these two different pieces that do come together in some ways.
But these grants, they're still very the process for deciding them is very conversational. It's very much based on what's the impact of the organization. What is the opportunity to

actually be able to productively make use of a couple hundred thousand dollars, which is not much when you think about the scope of the problem.
And is this a, a group who's gonna be accountable for how that's money is spent and report back? And really this is a multi-year kind of relationship that we can build with them. It's a lot of this stuff, [01:05:00] again, boils down to humans talking to humans.
Right,
Eric: excellent. So, to break down a little bit more about what the open source security foundation does, we talked a little bit about education. We obviously talked about the Alpha Omega project. We talked about the mobilization plan, focusing on vulnerability discovery, shortening ecosystem patch response times, OSS production code audits. What other things have we missed in, in identifying the scope of what you do? What big items do you think?
Brian: Looking at something like Zig store is not just software. Zig store is also a service. Zig store is also a kind of a standard a specification of the very least for how these pieces of signing infrastructure and the clients should actually interop operate.[01:06:00]
And so as a project at the Open ssf, we are set up to try to tackle all three of these different angles. Obviously supporting open source projects that write code is super easy. That's something Lin Foundation has been doing for 20 years. Developing of specifications and standards that can be, considered reference quality is different than writing open source code.
It requires a different IP management kind of framework. It requires thinking about how do we get the stuff, part with. Organizations that can then work their way into regulations. So for example we have a relationship with iso where if we follow a certain process for the development of a spec, we can then submit that as a ISO standard using the publicly available standard process that they use.
So we've done that with s Spdx, which is an ssam format. We'll probably do that're on a path to do that with a couple of other things coming outta the open ssf. So this kind of development of specifications and standards to try to get the [01:07:00] globe kind of converging on the same set of security practices is really gonna be essential when you think about just how many different pieces are in the average enterprise offer stack or frankly, in the average cell phone.
These things are global. They cross lots of different language communities and we need things like six doors to be adopted, very horizontally to be able to get efficiency in economies of scale. So spec work To some degree also being able to run an operations around something like the six store key server or in the future, other resources that would be basically look up services or ways to understand risk and open source projects.
Again, using open data using practices that map. Open source communities tend to work, but going beyond the office of code and finally making all these pieces come together in kind of a unified hole. It will be a big focus for us this year. We, we have a home run [01:08:00] with SIG store.

We have stuff that's still emerging like salsa and S two, C two F. Our, our monitoring tools as well are individua, our risk measurement tools are things that you deploy one by one, but figuring out how to bring those all together into a unified thing that can be plugged into C I C D pipelines to help essentially the, the system, the build systems of the world, and become, little secure software factories, is something we'd like to see.
And that means painting the picture of how these pieces work together. Finding ways to get them implemented and picked up by, by the major distribution sites out there, Maven Central and pm pi. And then everybody talking the same terms in terms of how this stuff gets out there and widely adopted is a much more monumental task than writing software. But something that we've gotta do if we want to try to secure the whole of the open source e.
Eric: We've got your work cut out for you. But it sounds like you've got a [01:09:00] great start. And more before we break anything else we should touch on before, before we do.
Brian: That's really the expanse of what we're working on at the op at the open ssf. But most importantly, we are, we're made of people . We're like soil and green. We really depend upon folks to not only use the stuff that we're creating and organizations and but also to come and participate.
And so if you're a an open source developer, if you're a cybersecurity expert, if you're building tools in this space as a company and you want to make sure you're building on top of what everyone else is building on top of rather than competing against the where folks are heading, please come and join our community.
We are incredibly open. Come to the website. There's great links about how to get involved, links to our Slack channels, links to our mailing list, links to our Zoom calls. We're a very chatty bunch, but also a very welcoming bunch. So please jump in the water's fine.
Eric: Awesome. Thanks so much for coming on and telling us about it.
[01:10:00] And you're doing great work, so thanks so much.