Sonatype Nexus Lifecycle Review From A Customer

IT Central Station
Unbiased reviews from the tech community

Sonatype Nexus Lifecycle

Review by a Real User

Verified by IT Central Station

Security Team Lead at Tyro Payments Limited

EdwinKwan

WHAT IS OUR PRIMARY USE CASE?

It's mainly used to scan for security issues in any components that we use. There are two parts to it, the license part and the security part. We use it generally for the security, but we also do have scans for the license stuff too.

HOW HAS IT HELPED MY ORGANIZATION?

One of the ways that it has helped us is that it has given us visibility into security issues. It has made us a bit more proactive in dealing with things. Before, we depended on how much news there was about a particular issue in a component, just learn about it. And when we learned about it, we didn't know which applications we had that were affected by it. Lifecycle helps really well with that. We put it into our pipeline. Whenever a developer builds, he can choose to do a scan - we don't enforce it. But what we do enforce is that when a developer makes a change in the repository, which means pushing it into production, as part of the build pipeline we scan it to make sure they are not introducing anything new in there. That has been a really good feature to make sure we've got that base level of hygiene. It also has something called continuous scan. We run that every night and scan our build materials - all the components that we know we are using, based on the previous scans. We re-scan them to see if any of them have any new vulnerabilities that have been detected. That is really beneficial because in our company we're always building new applications, and some of them are more actively developed than others. What we found was that we had a lot of vulnerabilities in applications that weren't being actively developed, things that needed to be fixed. If it weren't for Lifecycle, they would have just fallen off our radar. It has brought open-source intelligence and policy enforcement across our SDLC. We have two kinds of build pipelines. They are centrally managed by a team which handles all the build infrastructure. We integrated it so they have to do those scans. The policy enforcement will break a build, so you can't move forward without addressing it. The solution blocks undesirable open-source components from entering our development lifecycle, based on the policies that we set. It will break the build straight away. There's no way you can ship code that introduces new vulnerabilities. We just don't allow it at all. It has improved our security but, in terms of developer productivity, if you asked the developers about fixing security issues, I don't know if they would consider that productive for them. But from my point of view, it has improved developer productivity.

Sonatype Nexus Lifecycle

Read 10 reviews of Sonatype Nexus Lifecycle

WHAT IS MOST VALUABLE?

There are two things that allow us to do what we want to and that's why we chose Nexus Lifecycle. First, it scans and gives you a low false-positive count. When we were looking for a product to solve this need, we looked at different products, Nexus Lifecycle being one of them. The reason we picked Lifecycle over the other products is, while the other products were flagging stuff too, they were flagging things that were incorrect. Nexus has low false-positive results, which give us a high confidence factor, which is something we like about it. The other thing that we thought that was really good about it was that it gives an overview. We find something that has a vulnerability and say, "Hey, what can I upgrade to?" What's really nice about that is it shows us a graph of all the versions for that particular component, and it marks out the ones that have a vulnerability and the ones that don't have a vulnerability. It also shows the popularity, so we can look at it and say, "Alright, from where we are, what is the next version that we can move to that is not vulnerable and that is quite popular?" If it's popular, we tend to prefer it because then more people are looking into it, and it gets a bit more scrutiny.

WHAT NEEDS IMPROVEMENT?

We created a Wiki page for each team showing an overview of their outstanding security issues because the Lifecycle reporting interface isn't as intuitive. It is good for people on my team who use it quite often. But for a tech engineer who doesn't interact with it regularly, it's quite confusing. We did that because we got so many questions about it all the time. There are other areas for improvement. The most recent one - something I haven't shared with Sonatype yet but I intend to - is with the creating of defect tickets. The solution has something that is really useful, its integration with JIRA, and it creates tickets if there's an issue. What I thought would be really good was, from the moment we break builds, there is no way to track, from a management perspective, how we are doing. We are looking at creating tickets. The problem with the tickets, which is the where there is room for Sonatype to grow, is that there is no flexibility in terms of customizing the entries in the tickets. There are certain things they put in for you, they tell you what application it is, but what I'd really like to be able to do is say, "Fill in this field with the name of the application. Fill in this field with the name of the owner. Or set a due date to be X days from when it was raised. They don't allow that. They allow hard-coded values across everything in Nexus IQ. It doesn't work well because the tickets created depend on the use case. We would like to create these tickets and give them directly to the teams that have to look after them. We want to be able to assign them to the right person, based on the application that is used. " We are looking at finding ways to integrate with it because they don't have that. Another feature they could use is more languages. Sonatype has been mainly a Java shop because they look after Maven Central. And we have been mainly a Java shop in development. But we've slowly been branching out to different languages. They don't cover all of them, and those that they do cover are not as in-depth as we would like them to be. They don't have the same level of coverage as the main language, which is Java.

FOR HOW LONG HAVE I USED THE SOLUTION?

Three to five years.

WHAT DO I THINK ABOUT THE STABILITY OF THE SOLUTION?

The stability has been pretty good. We're pretty happy with it. There have been no issues there.

Sonatype Nexus Lifecycle

Read 10 reviews of Sonatype Nexus Lifecycle

WHAT DO I THINK ABOUT THE SCALABILITY OF THE SOLUTION?

Scalability is not an issue. We have a microservices architecture and we've got about 150 applications in there and we scan them quite regularly. When we first started, we had a lot fewer applications, we were sending about five gigs of scanning data requests to the Sonatype servers every day. They were able to handle that. We had issues before, but I think they were more networking configuration issues, and they could have been on our side. But that has all been resolved and there are no issues.

HOW ARE CUSTOMER SERVICE AND TECHNICAL SUPPORT?

Their technical support has been pretty good. They're based in the US and the turnaround time tends to be overnight. We generally send out requests in the afternoon and we normally hear back from them when we come into the office the next day. The depth of the responses vary, although it's generally pretty good. Sometimes they just don't have enough information, and that could be that from our side, that we have not provided enough information, enough context. But generally, it's been alright.

HOW WAS THE INITIAL SETUP?

We had a few issues initially when we set it up. We had a problem with not having enough space because it would keep the reports indefinitely. We were running out of disk space. But I know they've addressed that now because, in one of the updates that we did last year, the disk space was reduced considerably. They've been telling me that they were actively looking into it. The initial deployment took a few days. Most of the challenges that we had for the deployment were mainly to do with the rollout of our policies. Imagine an application that never had any scans, and we wanted to get to this SLA model, where you shall not introduce any more vulnerabilities and you need to fix existing issues. What took so long was we had to turn on the policies slowly and we had to grandfather everyone. Otherwise, everyone would just stop working straight away. When we first turned it on we discovered so many vulnerabilities in there that we never knew existed before. The implementation strategy was not to have the SLA initially - how long you had to get something fixed. We turned the solution on and said you can not introduce any more new components that have vulnerabilities. We drew a line in the sand and said, "That's it." Then, we created a list of all the things that we knew were a problem - that was a very manual process. We started from the top saying, "What are the critical ones that we will work on with teams to try and address them?" Some of the fixes were not trivial, they were quite a big change. One of the reasons was because, being an old application, it was using really old versions and the fix required a newer version. But the jump from where you were to where you needed to be was quite a big jump. That resulted in quite a lot of backward incompatibility with the other components in the system. That was what took a lot of time. We worked our way down. It took us a good year-and-a-half to get to where we wanted to be because we were competing with product engineering time to either work with features or fix security. We needed to find the right balance. For deploying it there were two people from my team to set it up and get it all going. And to address the issues it was a combined effort within the whole company. In terms of maintenance, now that it's configured, we have one person a week who is on the support roster to address any issues that we have. The maintenance is more to field questions the engineering team might have. They may say, "Hey, I just got this report that this application has an issue. Can I have more information about it?" Maintenance isn't about maintaining the system, it is more about providing consultation to teams and advising them on how to fix those issues that have been discovered.

Sonatype Nexus Lifecycle

Read 10 reviews of Sonatype Nexus Lifecycle

WHAT WAS OUR ROI?

The area where we've seen ROI is security hygiene. We're using a lot fewer vulnerable libraries. What we have seen is that when there is news about something that is vulnerable, and that there is a tool that someone has created that allows you to exploit it, we normally already know about it and we've addressed it. There's peace of mind knowing that we're on top of it.

WHAT'S MY EXPERIENCE WITH PRICING, SETUP COST, AND LICENSING?

We're pretty happy with the price, for what it is delivering for us and the value we're getting from it.

WHICH OTHER SOLUTIONS DID I EVALUATE?

We did a PoC with a few companies and we picked Sonatype and we've been happy with them since. We looked at Black Duck, and we also look at the free version, the OWASP, a dependency checker. We also looked at Veracode. The difference between Sonatype and the competitors is the accuracy. But having said that, I'm not too sure how Lifecycle compares to Black Duck. I know Black Duck is pretty good too. The main difference between Lifecycle and Black Duck for us was the price point.

WHAT OTHER ADVICE DO I HAVE?

My advice is that you should definitely use it. You need to think about the rollout and to make sure you integrate it into the software development lifecycle. That's where you get the most value because it provides quick feedback for developers. Be mindful of the rollout and breaking the builds. I don't think other companies that we spoke chose to break builds, but we do that and that is a sensitive topic for developers if you choose to do that. We don't use the application onboarding and policy grandfathering features at all. I suggested that to them, but the main reason we don't use them is, while we had that problem when we started out, we don't have the problem anymore. We don't use the Success Metrics feature as much. When it first came out I was quite excited about it, I thought it would be quite useful. But it hasn't really been as useful as I would have liked it to be. I was going to use it for figuring out trends. I was hoping to figure out how are we are tracking the number of vulnerabilities being discovered, and the trend, over time in terms of: Are we actively addressing them? I was hoping to break that down to engineering departments so could create a report and say, "Hey, this particular department has been really good, they're actively fixing vulnerabilities as they're coming out. This other department could be a lot better." I was hoping to get that, and it kind of had that. To be honest, I haven't looked at it for quite a while. But when I first looked at it, it looked quite good, but I didn't understand quite a bit of the graphs. I ended up using my own data set instead. We do have metrics on how much faster it helps us to fix issues but that's more because we have a company policy, we have an SLA there. It's based on the severity of the issue. There is a CVSS code. We map that into criticality, so if it's a ten, we say it's a severe security issue. There are ranges: critical, high, medium, low. This is actually mapped out to some standard policies that come with Nexus Lifecycle when you first install, so we just kept that in there because we thought that was best practice. But what we did say is that if there is anything that's critical, we want the team that's looking after the application to immediately stop work and address it straight away. If it's a "high," they have one month to address it. If it's a "medium," they've got three months, and if it's a "low," they've got six months. That's how we choose to address it, but that's set by us and it's enforced by Lifecycle. We have done something to integrate with it. It's not part of the feature set that it has. We integrated with it such that when we do discover something that's new - nothing that's introduced; rather something that's already in there that was okay yesterday but isn't okay today - we put a policy waiver (which is the term they use in Lifecycle) in place so it doesn't break the build. Once that SLA has expired, it will break the build and teams cannot make any more changes until they address it. That helps us conform to the SLA. The data quality is generally pretty good. We're pretty happy with it. We have seen a few cases in the last year where there were things that came out, and the teams came to us and said, "Hey, it's saying this, but we investigated further, it's not really an issue." So we've gone back to Sonatype and told them about these things. But, having said that, across the board, we feel that Nexus has been the most accurate so far, compared to all the other ones that we have used. It integrates fairly well with our existing DevOps tool. We had to do some work to get the metrics that we can show teams. We had to do some work to hand it the SLA stuff that we want our teams to go by. We are trying to do some work now where we want to create a defect ticket automatically. It hasn't been very good at that. It has some basic functionality but not as

good as what we want. But generally, I would say it's good. I would also add that I don't think that it's any better or worse than the other products out there. It's doing all right. The primary integration was to enforce our SLA. The other integration we have done is we created another tool that acts as a proxy. There are applications and applications belong to a team. It allows us to give immediate feedback to the teams. When the teams choose to build it locally and they run this tool, they don't use the Lifecycle tool, they use this tool that we wrote. The reason why we did that was for our SLA, because then the report comes back to the team. It actually shows them how many days remain for those things that are subject to the SLA. We also did some work to create a Wiki page, one for each team, that we update every day. This is more to give to team leaders, who are not always on the code, an overview of what the outstanding security issues are, in which applications they are found, and how much time they have to fix them. Regarding the time it takes to release apps, it hasn't changed the amount of time. We would like to move to continuous deployment but, at the moment, some of them are continuous and some are weekly and this has had has no impact on that. We have about 135 users of the product in our organization. Software engineers are using it, DevOps engineers are using it, we've got some testers using it. We also have some delivery managers using it and they're using it more for the reporting to see how things are going. We also have some operations people using because it can also scan containers. It has been utilized quite extensively. I don't think it's going to increase any more. It would increase if we had more applications, but we are also using a lot more technologies. I give it a nine out of ten because of the accuracy. I like the information that it provides in terms of how to address issues. It would have been a ten, but there are other things that require integration, the extra stuff that we had to do, which I wish we didn't have to do, that it was all done for us. But we're probably not using it in a way that they envisioned most people would use it.

Read 10 reviews of Sonatype Nexus Lifecycle