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

Configuration Manager at a health, wellness and fitness company with 5,001-10,000 employees

ConfigManag73548

WHAT IS OUR PRIMARY USE CASE?

Our primary use case is preventing major security vulnerabilities. We use it as part of build our pipeline. We have a plugin that gets scanned by Sonatype as the build runs and it scans for all third-party dependencies. We haven't yet gotten to the point where we fail a build, but we make the matrix visible so we know where we need to focus. In the coming months, we plan to actually start failing builds and preventing releases which have certain vulnerabilities, from going into production.

HOW HAS IT HELPED MY ORGANIZATION?

One of the ways it has improved the way our organization functions is that it has created awareness of unlicensed, third-party dependencies and insecure vulnerabilities. That's what it has done at the moment. It's going to have a bigger impact when we start enforcing it on the build pipeline. Then they will be at a point whereby they have to conform to the policy. There are a number of open-source components that it has highlighted. It means that we have to find a replacement for them. There's a recommendation that's given when it highlights them. So we can remain in the same open-source components, it's just that there has been a patch or an update to them to close off vulnerabilities. In the context of remediation costs, there is the issue of reputation if you are not releasing secure apps to market. That's one major problem. Secondly, if you know you're releasing insecure applications, you're incurring technical debt because you're going to have to, at some point, mitigate those security vulnerabilities. It's become a mandatory control now. We just went through a process of defining all the DevOps controls, and one of those controls is to use Nexus IQ. They can't skip that part of their build pipeline. Every code has to be scanned.

WHAT IS MOST VALUABLE?

There are a number of features that we find valuable. The basic functionality of Sonatype is its scanning feature. Out of that, you get the reporting capability as part of your build and it gives you the statistics as part of the build report. There's also a feature whereby, in your IDE, you can get immediate feedback as you're developing. That's also quite a handy feature. In addition, in our Nexus repository - we have Nexus OSS and Nexus Lifecycle linked together - all our third-party dependencies are scanned. The policy management is quite federated, in a sense, whereby we can assign a policy specific to an application. The grandfathering mode allows us to add legacy applications which we know we're not going to change or refactor for some time. New developments can be scanned separately and we can obviously resolve those vulnerabilities where there are new applications developed. The grandfathering is a good way to separate what can be factored now, versus long-term technical debt. In most cases, these legacy applications are simply retired, so to refactor them wouldn't make sense. Most of them go through a rewrite cycle or are replaced by something we have purchased. There's a very interactive view where there's a recommendation, as part of the reporting. You can click on a certain vulnerability and it will give you a recommendation. For example, if you're using something that's not licensed or has a certain license type, it will recommend to you, "You should go onto this license," or, "Go to this version, which covers this vulnerability." There are actual recommendations that are synchronized with the database in the

States. The solution integrates well with our existing DevOps tools. We're using Atlassian and using Bamboo as part of that Atlassian set. Bamboo is our continuous integration tool. There's an out-of-the-box Nexus IQ plugin for Bamboo. It's really simple to configure and it gives you results as you're building. Also, the API is very rich, meaning that we don't necessarily have to get a report from the front end. We can build custom reports through the API.

WHAT NEEDS IMPROVEMENT?

They have recently released some online training documentation, because we had to a lot of our own learning. If they had a more comprehensive online tutorial base, both for admin and developers, that would help. It would be good if they actually ran through some scenarios, regarding what happens if I do pick up a vulnerability. How do I fork out into the various decisions? If the vulnerability is not of a severe nature, can I just go ahead with it until it becomes severe? This is important because, obviously, business demands certain deliverables to be ready at a certain time. So, some online tutorials around the administration, around developers, on how to use the tool effectively, would be a good idea. Where we have extended is that we're using third-party Docker containers. So, we're not just using third-party libraries, but we also have third-party-type Docker containers, or containers from Docker Hub, for example. Somebody else has built the Docker image and we're using the Docker image. Scanning of security and vulnerabilities on that image, specifically, would be useful. It would be good, as we're building an image, or as we're running an image, if we could decompile that image and scan it, to look for any vulnerabilities or any areas where there's been a violation of licensing. For example, we could download a WebLogic container, which could be an infringement of licensing. It's things like that which our developers need to be mindful of. They could simply download any container from Docker Hub, without being aware of the licensing violations or the security vulnerabilities.

FOR HOW LONG HAVE I USED THE SOLUTION?

We've been using it for about two years.

WHAT DO I THINK ABOUT THE STABILITY OF THE SOLUTION?

The stability is very good. It's extremely stable. We haven't had an instance where it's running out of memory or anything else.

WHAT DO I THINK ABOUT THE SCALABILITY OF THE SOLUTION?

We haven't had an instance where we have run into such high volume that we needed to scale. The only change we made was to increase memory, because we started utilizing the API. In terms of redundancy, all the data is sitting in the database. We have backed up the folder structure, and the worst case is we just restore that folder structure onto any server. You could run it in Docker if you wanted to, as well so that is immutable. It's been made to be a lift-and-shift type of product. We have 100 users actively using it at the moment. They are developers, mostly.

HOW ARE CUSTOMER SERVICE AND TECHNICAL SUPPORT?

We haven't had an instance where we needed to call tech support. There haven't been issues to the extent that we needed to get hold of tech support. Most of it we deal with in-house. The last issue was probably a year ago, where we ran out of memory and that was a simple config change.

IF YOU PREVIOUSLY USED A DIFFERENT SOLUTION, WHICH ONE DID YOU USE AND WHY DID YOU SWITCH?

We did not have a previous solution. We brought on Nexus Lifecycle because there has been a heightened, more aggressive stance on security.

HOW WAS THE INITIAL SETUP?

The initial setup is pretty straightforward, both the installation and upgrades. We're running it on a Linux environment, so there's not much that we needed to do there. Policy management is probably where it gets a bit complex. That's where I referred to the need for some tutorials, some more comprehensive documentation. The initial deployment took less than an hour. It's very quick. Download it and run the .jar and you're good to go. We do use an Oracle Database, so we did need to set up the database. We just pointed it to one of our Oracle instances. The implementation strategy was to start enforcing the policies and, eventually, to prevent things. We are implementing it in such a way that the developers will, at the point of development, in their IDE, be faced with these warnings that they are using insecure, third-party dependencies and where they are violating the licenses. That's the strategy, whereby we simply block such releases from getting into production. That's where we aim to get. For deployment you literally just need one person. For maintenance, we have just two people, and that's for an entire pipeline. They're automation

specialists so they provide automation solutions. They also act as application administrators.

WHAT WAS OUR ROI?

We haven't seen ROI as yet, simply because we haven't been harnessing the full potential of the tool. The way I think we will potentially see a return on investment is if we are using any licenses that could be costing us indirectly. We could be looking at certain technical debt which we could be dropping. Those are the aspects we could look at, but we haven't yet maximized the true, full capability.

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

Our licensing is bundled. We pay a single licensing cost for both Nexus OSS and Nexus Lifecycle together. So I'm not sure what the individual costs would be. We bought both Nexus Repo - we're using Nexus 2.0 and Nexus 3.0 - and Nexus Lifecycle.

WHICH OTHER SOLUTIONS DID I EVALUATE?

There's SonarQube which does static code analysis, but not at the level that Nexus IQ offers it. There is Artifactory, which does do Docker scanning now. One thing that Nexus IQ has been able to do is to be almost proactive in its integration. You can be in your IDE, you can be in the build pipeline, you can be in the Nexus Repository, and you can get a view of the vulnerabilities. Also you can get recommendations, so you don't necessarily have to waste time in searching the web for a patching solution or an update to fix the vulnerability. It actually gives you recommendations about what you can do to mitigate the problem. That's a distinguishing feature from the other toolsets.

WHAT OTHER ADVICE DO I HAVE?

Have a key, a defined goal because, as much as the tool is there, it isn't able to create a goal. The goal is, "We would like to improve the security of our codebase by at least X percent, ensure that 90 percent of our applications, for example, going to market are secure applications." With that goal in place, I would look at purchasing the tool because it would be an immediate implementation of that strategy. We bought the tool with that idea in mind, but nothing clearly defined on a granular policy level. But that's ultimately what makes the difference, to say we are focusing on looking at these types of either licensing or dependencies. The tool is then just automating that process or enabling us to do that. It's taken us about two-and-a-half years to implement a strategy. Whereas, if we had the strategy initially in place, it would have gone much faster. The governance is centered around security and licensing. Those are the core governance factors around the policy. From a dev SecOps perspective, it's fundamental for governing your third-party dependencies. That's where the enforcement comes in. Once you've defined the policy, it becomes the law, and you can enforce that law. If you are working in a SecOps-type environment then you would sign off on that policy to enforce that. We haven't used the Success Metrics to its full potential. We understand it revolves around targets that need to be set and how far you are from those success targets, but we haven't used that as yet. I wouldn't look at it as a root-cause analysis or a monitoring tool. Yes, it does scan for security vulnerabilities and where we've violated licensing, but it's not used as an Elasticsearch, a Splunk-type, or Dynatrace-type of tool. When it comes to troubleshooting and root cause analysis, will we use our Dynatrace and our Elasticsearch to look at the logs. A lot of the third-party dependencies are open-source dependencies. A lot of them are provided through Apache, etc. We always look at enterprise solutions because of the nature of our business, but where the is an open-source piece of software which we can utilize as part of our enterprise solution then we do. We haven't scoped Nexus IQ to focus on open-source software. Nexus IQ in and of itself is commercial. If we had to use Jenkins or any of the open-source build tools, it would easily integrate with those open-source tools. Using Nexus Lifecycle, it might end up taking longer to release to market because you may need to refactor. In an industry where these applications have already been developed and now you end up scanning them and you pick up all these issues, you are going to have to refactor them. That means that either new requests must go into a backlog or you fix technical debt, which is what Nexus IQ is going to tell you to do. It might have an adverse impact at first, but the goal is to get to a point where you break even. Now we are at a point whereby our apps are being released into production as secure applications. There is the opportunity with new applications, from the onset, to ensure that they are released as secure apps into the market. That doesn't mean that they might be released faster because that could depend on a number of factors. It could depend on your testing cycle, it could depend on your release process, etc. We haven't had a one-on-one sit-down, or a survey done to really gauge the type of developer engagement. Our level of adoption has been a little inconsistent. Right now, they're getting the visibility of these metrics, but they don't actually have to do anything about it. But once we enforce the policy that their builds and releases will fail, then they will be forced to do something about it. From a SecOps perspective, it enables them to put that application on a risk register and say, "Listen, we need business to focus on fixing this application, making sure it's secure." That's the direction we are currently headed in. Developer

productivity could be based on how automated our pipeline is. We are there. We just have to focus on dev and nothing else. This might actually give them that awareness.

Read 10 reviews of Sonatype Nexus Lifecycle