Fortune 100 Company Speeds Up Discovery and Reporting of Log4j Vulnerabilities

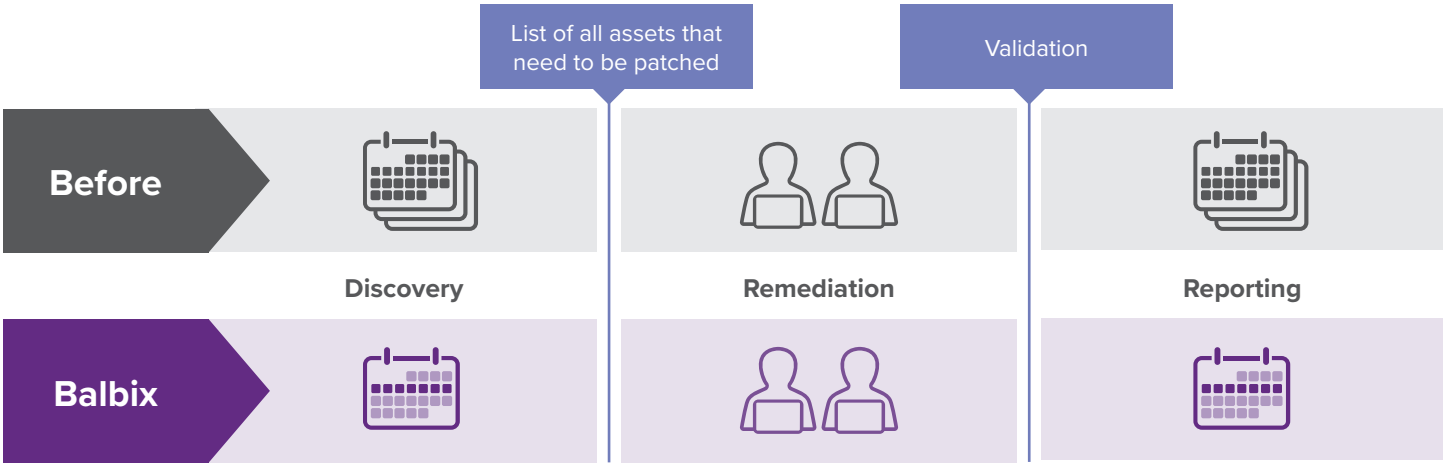
The Challenge

Log4Shell, a zero-day vulnerability in the Java logging tool Log4j, was publicly disclosed on December 9th, 2021. The following day, the Balbix research team was contacted by one of our customers, a Fortune 100 company, for assistance with finding and patching Log4j instances amongst the millions of assets they manage. At the onset of the crisis, they estimated it would take 2-3 months just to discover instances of Log4j across their environment, and several more months to remediate instances that were vulnerable.

Enter Balbix

With the help of the Balbix platform, the infosec team significantly compressed that timeline from months down to weeks to meet their internal mandate of identifying and fixing all identified instances in 6 weeks. This prompted their very relieved director of application security in charge of the project to breathe a sigh of relief and say, *"I should have some free time back now!"* He also added, *"everyone needs tools like Balbix that can look at the devices but also see into the software stack."*

Reduce Log4j Discovery and Reporting from Months to Weeks



Balbix was able to reduce the time for discovery and reporting from months to weeks

> *"Everyone needs tools like Balbix that can look at the devices but also see into the software stack."*

Kicking off the Initial Search for Log4j Instances

Finding Log4j instances is complicated by the fact that it can exist as (1) a library resource, (2) a JAR file in a third party application or (3) embedded in a custom application. When Log4Shell was publicly disclosed, this customer identified 2700 instances of Log4j that resided as an independent application (not as an embedded JAR file) in their environment. Within hours it was determined that none were vulnerable to the exploit. This was the good news! However, rather than feel reassured, the team was skeptical that their search was over. They assumed they had vulnerable instances elsewhere, an assumption that would quickly be proved correct.

The team put together a two-part plan to find the hidden instances. They would run their network scanner and they would use a scripting mechanism they had previously installed on all servers to scan their drives for the Log4j name or file hash in order to identify the presence of Log4J.

Unfortunately, the network scan results proved burdensome to use. A quick review showed an equal number of false positives and real instances of Log4j. The scan mechanism also was problematic as it took too long. For example, one server scan was only 40 percent complete after running for 48 hours. Plus, despite running the search at an appropriate *"nice value"* to avoid affecting the other resources requiring compute power, the security team got complaints from IT and users. The security team estimated it would take 2-3 months if it depended on these tools to discover their remaining Log4j instances.

Quickly Discovering Instances by Using Balbix to Examine Run Time Processes

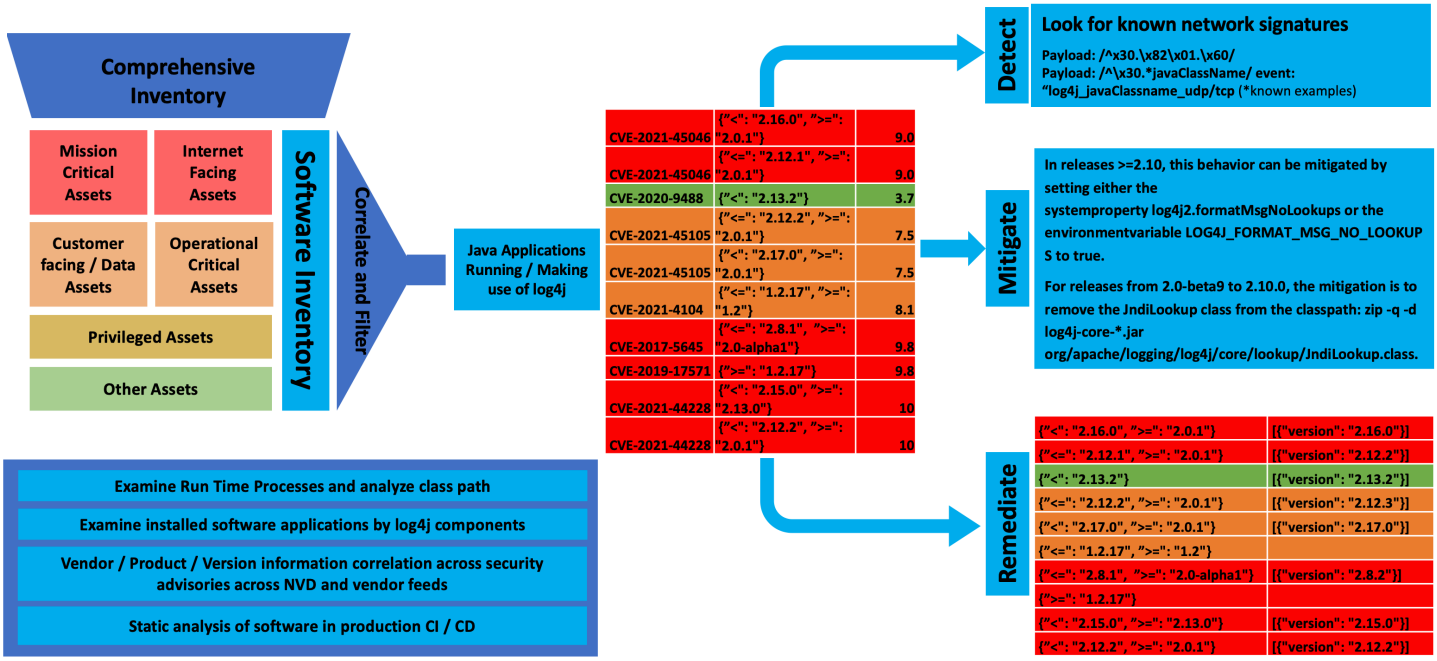
The team at Balbix set out to help this customer find runtime instances of Log4j. Balbix already had a tremendous amount of information about the customer's environment since providing a continuously updated asset inventory is a foundational part of the Balbix platform.

As a result, Balbix already had most of the needed information. Since Balbix determines asset types, we knew which devices were servers. For those servers, we also knew:

- ☑ what software was deployed
- ☑ the asset owner
- ☑ the person responsible for managing IT changes
- ☑ and hundreds of other attributes.

What was missing was additional fidelity about the third-party applications running on those servers.

The Balbix engineering team was able to make some quick improvements to gather that data. For example, we ingested data about their Linux systems using (1) RPM Package Manager for Red Hat distributions and (2) Debian package manager for Ubuntu and some other Linux distributions, in order to discover dependent Log4j packages. To detect cases in which Log4j was not installed via package managers, as is often the case for custom applications, Balbix successfully used live process analysis to analyze libraries and then correlate and deduplicate the results. The information we gathered included the affected application, the application version, where they were installed and the full path and version of the Log4j components.



Overview of the process used by Balbix to detect, mitigate, and remediate Log4j CVEs

Within hours Balbix was able to start identifying new Log4j instances, eventually identifying Log4j embedded in over 1000 separate application versions across more than 60,000 assets, including 450 application versions on approximately 5000 critical assets. The security team told us that the *"data from Balbix is highly accurate and actionable—4x to 8x better than the information obtained from other sources."* Also, the false positive rate was lower than 0.1%.

> *"Data from Balbix is highly accurate and actionable—4x to 8x better than the information obtained from other sources."*

Rapid Prioritization and Remediation

Balbix was then able to draw on other attributes we had pre-mapped for the servers to provide the business context of the affected assets in order to prioritize remediation. As a result, the security team at our customer quickly prioritized remediation based on factors such as:

- ☑ Whether the server was business critical
- ☑ Whether it was internet-facing
- ☑ Whether it held customer data

With a list of assets to remediate in hand, the director of application security at the customer would then normally reach out to each application owner to have them deploy the required patches. To determine whether the task was completed, the AppSec team would then have to keep reaching out to the application owner to ask if the update was completed, or they would have to run a scanner. This would be a laborious and iterative process.

Instead, they were able to save a lot of effort by tracking mitigated applications in Balbix as Balbix continuously updates the status of every asset including the software version and Log4j mitigations (including environmental variable changes or the removal of classes from the classpath). The team still had to follow up with application owners who had not completed remediation in a timely manner, but it was a much shorter list of tasks, and allowed them to feel confident about meeting their mandated deadline of fixing all identified instances by January 17th.

For each instance, Balbix also provided the appropriate patch version for each instance within the platform (see summary table below). The customer noted that, *"the fact that we are able to detect the mitigations is icing on the cake which helps us to reduce the communication overhead with 1000s of application teams."*

Vulnerable Applications with Critical / Vulnerabilities with RCE

CVE	Vulnerable Versions	Available Fix	Severity
CVE-2021-45046	{"versionEndExcluding": "2.16.0", "versionStartIncluding": "2.0.1"}	[{"version": "2.16.0"}]	9.0
CVE-2021-45046	{"versionEndIncluding": "2.12.1", "versionStartIncluding": "2.0.1"}	[{"version": "2.12.2"}]	9.0
CVE-2020-9488	{"versionEndExcluding": "2.13.2"}	[{"version": "2.13.2"}]	3.7
CVE-2021-45105	{"versionEndIncluding": "2.12.2", "versionStartIncluding": "2.0.1"}	[{"version": "2.12.3"}]	7.5
CVE-2021-45105	{"versionEndExcluding": "2.17.0", "versionStartIncluding": "2.0.1"}	[{"version": "2.17.0"}]	7.5
CVE-2021-4104	{"versionEndIncluding": "1.2.17", "versionStartIncluding": "1.2"}		8.1
CVE-2017-5645	{"versionEndIncluding": "2.8.1", "versionStartIncluding": "2.0-alpha1"}	[{"version": "2.8.2"}]	9.8
CVE-2019-17571	{"versionEndIncluding": "1.2.17"}		9.8
CVE-2021-44228	{"versionEndExcluding": "2.15.0", "versionStartIncluding": "2.13.0"}	[{"version": "2.15.0"}]	10
CVE-2021-44228	{"versionEndExcluding": "2.12.2", "versionStartIncluding": "2.0.1"}	[{"version": "2.12.2"}]	10

Summary table based on in-product mapping of fixes to vulnerable versions

The inventory also included a list of known vulnerabilities for each asset. This data proved especially valuable as new Log4j CVEs were identified in the days following the first disclosure. There was no need to re-run scans to identify the new vulnerabilities released by Apache. Application owners were able to combine their response into one set of actions to address all Log4j CVEs, and the security team was able to track all Log4j CVEs in a single report.

Postmortem: Benefits

With the benefit of hindsight, easily the most important benefit our customer saw was the ability to compress time. By having an accurate asset inventory in place at the time of the Log4Shell notification, the team was able to work with Balbix to quickly discover Log4J instances and track completed updates. Having the asset level details also improved the value of their network scanners when they were used in combination with Balbix. As our customer noted, *"accuracy, specificity and actionability have been the key factors in tracking this vulnerability and remediations/mitigations. With Balbix's help, [we] reduced the end-to-end risk mitigation workflow time from months to weeks."*

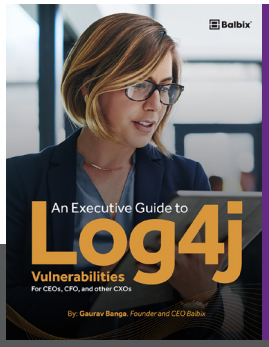
The team was also able to reduce the skill level of the responders. With much of the discovery and tracking being done through automation, the senior members of the team were able to delegate many of the remaining tasks to their junior teammates.

> *"Accuracy, specificity and actionability have been the key factors in tracking this vulnerability and remediations/mitigations. With Balbix's help, [we] reduced the end-to-end risk mitigation workflow time from months to weeks."*

Overall, the response to Log4Shell highlighted the benefit of automating CVE management—discovery, prioritization and remediation. Specifically, it showed the power of correlating all relevant information. The effort to prioritize Log4j remediation would have been considerably more difficult if some of the asset attributes were unavailable. For example, what would the team have done if the inventory didn't list which servers were internet-facing?

The team used the analogy of a fishing boat on the water to highlight the importance of their improved visibility. Many tools can tell them what is on the surface of the water, they said, but only Balbix could show them what lived below the surface. Continuing the metaphor, they felt that Balbix could identify every fish, every octopus, all the way to the bottom.

Finally, the Log4Shell incident reinforced that security posture management is an area of executive visibility. As part of their response to Log4Shell, the team had to provide a report to the CISO on a daily basis, and the CISO in turn, had to report to the CTO. With that kind of visibility, reporting on vulnerability management is one part of cybersecurity you definitely want to get right. As we wrap up the Log4j project, our customer has reinforced that reporting is another area where Balbix can provide critical assistance: *"we will be relying heavily on Balbix as the teams return to full strength. I want to make sure I don't get in front of the headlights as we craft our officer communications."*



To learn more download our **Executive guide to Log4j vulnerabilities** eBook