Heather:

Welcome to the Hurricane Labs Podcast. I'm Heather, and today we're going to be talking about Log4Shell. So joining me, I have Tony, Josh, and Kurt. Thanks, guys, for taking some time, with all of this craziness, to talk to me today. Let's go ahead and dive right in. So CVSS rated this vulnerability as a ten out of ten, which sounds like it might be a little bad. Why is this? What's going on with this vulnerability? Why is it that this is so serious?

Tony:

First off, the attack surface is gigantic. Do you know how every time when you go to install the latest version of the JRE and Java threateningly tells you that there are billions of devices running Java? This is one of those cases where there are billions of devices running Java. And a lot of them in enterprise environments, where they're running critical applications that we have to be responsible for, are using Log4j2. So it's this open-source logging library that was developed by the Apache Software Foundation and is maintained by a handful of individuals who were doing it as a charity. Or, most open source applications are doing it for the fun of it. But it has more or less become the logging standard for Java applications. So the issue that we're seeing in Log4j2 has to do with functionality called lookups. It's a way for Java applications kind of to refer to variables or insert certain formatted data into the logs. And in particular, this vulnerability is most concerned with JNDI lookups, which is the Java Naming and Directory Interface. It's a way for lookups to make requests through different protocols, the most common ones being LDAP and DNS, but there are a couple of others out there. So we have this massive attack surface; we have this functionality to do DNS lookups. What else can be done with that? Well, depending on what version of Java you have running, you can run a malicious LDAP server. And you can tell the Java application when it goes to do the LDAP lookup, "Hey, I have this class file over here that will totally help you get what you want; it's definitely not malicious, trust me," and it will go ahead and load it. In other cases, when you do these DNS requests, the attacker could stand up a DNS server, and you can format the JNDI request to send environment variables with that DNS request. So it's a part of the DNS query. So if the attacker owns a domain and sets up an authoritative DNS server for that domain, they see those queries. So in Cloud environments, especially where you have environment variables that define secrets that define user names and passwords. That's a big deal because now they're just logging your credentials in your Cloud instances. And then, aside from the DNS issue and the malicious LDAP issue, you also have problems where different applications have things called gadgets in them. When you set up a Java application, there are libraries, and libraries have a variety of functions, of course. And some of these functions might not be a part of the application you're using, but they're still there, and they can be leveraged by attackers to get code execution anyway. So that's where the big deal is, that you have all of these different methods of exploitation, in combination with a massive attack surface.

Josh:

Yep, and that's what makes it harder to detect, too, because you have those multiple different methods. So there's a lot of potential for obfuscation with this from using environment variables.

Tony:

Oh yeah, definitely. That's another thing that a lot of people are putting in. I want to say the Sophos Blog mentioned it; Cloudflare's blog mentioned it when they were talking about exploitation attempts. It's really hard to formulate proper queries and block lists for this because it's very easy to create workarounds and obfuscate the JNDI lookup.

Heather:

How are we seeing it being exploited right now?

Tony:

Well, again, I've seen a couple of researchers posting instances where attackers are requesting AWS secret access keys as a part of the lookup. And then, in other cases where you have Java applications that are running old versions of the JDK, it can just be told to go pick up a malicious Java class file, and that gives you code execution. Those are the primary methods I've been seeing.

Josh:

Yeah, and I've usually seen the scanning coming from HTTP user-agent headers, sometimes API keys, and the payloads in those mostly being coin minors at the moment, but I'm sure that'll evolve.

Kurt:

I already saw a few things for ransomware spawning from all of this, as well. So I'm sure it's just a matter of time, Josh, until it starts getting really messy.

Tony:

The scariest thing about all of this is that there is a variety of different ways you can go about attacking it. And for the most part, we're monitoring network activity, those HTTP headers, and URI strings. But potentially, anywhere you can input data into a Java application, and it's logged by the vulnerable library, that's an attack surface. And an example that I was in a talk with a couple of other security researchers over social media; let's use a really strange example and say, what if you had an aproxcard. You decided to encode the JNDI string into your card; you go to scan the card reader against your card, and the physical access system is running a vulnerable version of this library, and you just exploited it. It's a weird out-there attack surface, but it's still valid because if it's running Java and it logs that access attempt, then it just does whatever the JNDI lookup tells it to; that's a potential issue.

Josh:

Yeah. Wherever you're logging user input, you have the attack surface.

Kurt:

That's what I was going to say from more of a blue team standpoint. That's why writing detections right now and probably for the future is going to be nearly impossible. We're always going to be behind the attackers on it because we'll be finding the new encodings used, the different apps they're throwing it at. Regardless, you're going to sit there and cycle and be behind the gap of what we should be looking for, from actually a defensive standpoint.

Heather:

What should we be looking for? How do you know if this is being exploited on your system?

Josh:

If you have logs that have the JNDI string or some obfuscated version, and you can pull out an IoC out of that, like a domain or an IP, and you can find return traffic going out LDAP or one of the other protocols that are being exploited. That's how you can tell if something's been successfully exploited.

Kurt:

The problem that we're seeing, though, from a blue team perspective, is sometimes being a contractor and looking across multiple clients, logging into each client is different. And sometimes they have the logs or the firewall logs to actually see potential return traffic, where depending on what kind of host you're attacking, what software it's running, it may not even be getting logged for what you're trying to monitor with.

Heather:

Will this vulnerability be fixed, or is it something that is always going to have to be kept up with?

Kurt:

So, that's, probably another one of the gross parts. Yeah, that's probably one of more gross things about it is the fact that this isn't going away for probably years, for... We're still seeing EternalBlue way back when WannaCry became a thing. Still, in everyday activity and still, at actually being successful and exploiting, in my opinion, the Log4j stuff's going to be worse than EternalBlue. At least Windows was able to push the patches out, and it was only affecting Windows products. Now, as far as Log4j goes, as Tony said, it's anything that's running the Log4j2 library with Java, and you could use Java to program lots and lots of different applications. And the biggest thing noticed that some of the... For example, wasn't it, Josh you could recall, wasn't it, not CrowdStrike, but Carbon Black?

Josh:

Yeah, Carbon Black.

Kurt:

I know that they actually were vulnerable to the attack. And let's say if you're in a corporate environment using Carbon Black, that's out of your control. You're at... You're waiting for Carbon Black to fix it, and there's really not much you can do from your end, as far as waiting for the vulnerability to get patched. But on the flip side, you might also have software that some dev made in your environment ten years ago that's been outdated for the last eight, and you don't even know it exists. The attack surface is just massive. I could probably talk in circles more, but–

Tony:

... yeah and to like–

Kurt:

... I can't see it going away anytime soon.

Tony:

Yeah, I'd like to add to that, what he just said, it's the same deal. These are... Java's been used for an incredibly long time, and there are a variety of different verticals out there that might have Java applications as a part of their core workflow. They might rely on these applications and the people that

made them, the companies that developed the software for them. They might be out of business. They might have moved on. The developer might have moved on, or something might have happened to them. And then there's the fact that it's considered... You might have job applications in critical infrastructure as well, talking about power generation or other critical infrastructure out there. It's a big deal to get stuff patched in those environments because, you know, there's not real... You have to have multiple backup plans in order to be able to handle failures. If the patch fails to work properly, or if the application is updated and it doesn't work the exact way, then you need to have a detailed backup plan on what you're going to do to work around that. So a lot of the time, companies that are working in critical infrastructure or companies that have a particular workflow and no way to update the program without significant funds being put into it, they'll just opt to not updating it. So, that's, the other thing that's gross about this is that it's just... The term I've heard used before is it's a forever-day vulnerability. It's always going to be there–

Kurt:

... that's a good way to put it–

Tony:

... in those places that just don't have the option to update.

Kurt:

And I like the forever-day exploit because it's totally true, and then it's really not going away. And to add on what you said, Tony, that I think is really important, when you have critical infrastructure running per se, Java, that is vulnerable, if I'm not mistaken, you have to update the Java 8 plus at this point, even to patch for Log4j. And in that case, you might have to rewrite your code and make things actually... You might have to do complete rewrites on codes. So let alone having to make some small tweaks or something, that might take a long time for places that need 99.5% uptime.

Heather:

What about this Cybereason vaccine as a fix?

Kurt:

Are we not going to ignore how cheesy it is saying vaccinate now to a signature. I'm sorry, but that's so dumb. Okay. Now that I got that out of me–

Heather:

... tell us how you're really feeling–

Tony:

So the vaccine, huh?

Josh:

This is like a–

Tony:

Fix uses the vulnerability itself to set the flag that turns it off. So I wouldn't call this a vaccine per se so much as it's mitigation. And I've actually been doing a little bit of research on the mitigations out there. So what the blog post is suggesting is that they have a tool that will exploit the Log4j vulnerability. And then, it will either set the environment variable format message no lookups to true. Or, it will restart the program and use the Log4j2 format message in a lookups argument. And what that does is that it disables lookups, including JNDI lookups, entirely for your application. Now, on the one hand, that will fix the biggest issue of the LDAP server. Being able to tell your application, "Go download this totally legitimate dot class file that I have over here, trust me, it's not malicious." But the other issues of logging those DNS queries, or the other problem with the DNS queries, that's still an issue, regardless of whether or not you use this quote on quote vaccine. But the other problem that you run into is that the LDAP resource breath method that I'm talking about points it to that class file. That's only one of the types of attack available. I think I mentioned a little bit earlier, but the idea of there being gadgets in the application or portions of a library that could be utilized to get code execution; those are still going to be a problem. Even with this, or this mitigation method, that Cybereason is suggesting. So it's better than doing nothing, but the best method is, obviously, patching it. The mitigation methods that I've seen are using the format message; no lookups option, if it's available in your application. Updating the JDK version or the JRE that you're using to something that doesn't allow the... There's a feature called trust URL code base, and that's what allows the application to take that class file and download it. In older versions of the JDK, it's set to true, and it'll happily load whatever class file you give it. In newer versions of the JDK, that's set to false. Again, that doesn't really do anything if you are using the gadget method or if you found gadgets in the library are being used in your application to do code execution. So then you fall back to defense-in-depth techniques. That's looking for Java spawning, unusual processes, unusual files being dropped to disc, or being able to do egress filtering for your application servers. Or noticing that they're suddenly requesting doing HTTP request the port 8, 8, 8, 8 when that's not a normal thing that they do. So those are some more mitigation and remediation methods. It's just like we always say, with the blue team is that its defense-in-depth, it's layers of an onion. It's overlapping defenses.

Josh:

Yeah, and I believe this vaccine is also dependent on how it's being exploited too. Depending on where you're entering that string and how it's being logged in that application, it may not cover everything, just the most common.

Kurt:

So just to ask you guys, what would be your thoughts on the best way to find vulnerable instances of Log4j in your environment? I mean, at the end of the day, you do have to find where it exists internally, but is anyone–

Tony:

... it's going to be the unsexy answer that everybody hates, but it's being able to do asset management and software management in your enterprise. This is one of those things that's much easier to tell people to do, than it is to do it because–

Kurt:

... I'm going to be devil's advocate that, let's be the average company, that totally didn't do it; doesn't have it–

Tony:

... well–

Kurt:

What would be your steps?

Josh:

You could look for the file–

Tony:

... start with vulnerability, scanners and hope for the best. Honestly, though, that's probably the first answers. Your first line of defense is to take advantage of your vulnerability management program in internally. Or, using vulnerability scanners to at least test most of your web applications to see whether or not you get results that you weren't expecting.

Kurt:

I totally agree that having the assets all put together is obviously going to be your easiest way, but I just was being the... Almost everywhere I look, I've never seen, I think, a complete thing of assets at any client ever.

Tony:

I mean, and then you also have the point of view that a lot of these appliances and both security appliances and IT appliances, they might be running Java under the hood. And until you get that security advisory and you get whatever official patch that they offer for it, you're not going to be able to do anything with it.

Kurt:

That's fair too. I mean, I think that goes back to, should we keep running things that are critical, knowing that it's able to get exploited? Or, do we tear the thing down and try fixing it? But like you said, there's no patch available; you can't really do anything. Probably most people are going to choose to keep things running.

Tony:

Yeah, keep it running and make sure that it's being monitored as consistently as you can, until such a time comes that you can hopefully fix it.

Josh:

Yep, and restrict access where you can.

Tony:

Mm-hmm (affirmative).

Kurt:

I would sound a DMZ or some edit away from everything else. So if they do get compromised, you're not having–

Josh:

... well, that and this requires a public service to be affected. To be exploited from the internet that would require public service to be available. If you have something internally, that's vulnerable to this, and it's on the internet and doesn't need to be, implementing firewall whitelisting would be a good option.

Kurt:

That's a good point.

Heather:

All right. So let's talk a little bit about how this was disclosed. It was shared publicly on GitHub, right? So I'm wondering if identifying this vulnerability in that way didn't do more damage? It seems to me now it's like a sort of this crazy scramble where security teams are trying to erase hackers, patching versus exploiting, but it's also pretty pervasive. So I'm wondering, did a better way to disclose this vulnerability exist?

Josh:

Would've been nice if it was a Monday.

Heather:

Right, yeah–

Kurt:

... that's probably the most accuracy.

Heather:

And not right before a holiday.

Tony:

Oh yeah.

Kurt:

That always happens, I swear.

Tony:

That's always how it goes. I remember with the SUNBURST stuff; it happened right around this time, last year. At the beginning of this year, that was a mad scramble, as well. But like to plan your question there, normal disclosure is the vendor has 30 days to fix it. And then people will start looking at the patches since Log4j is an open-source project and it's on GitHub. You can just see the changes. They're like, "Huh. I wonder why they disabled lookups." And then everybody just goes, "Oh, ohhh. Okay!" But another point to make that we mentioned earlier, EternalBlue has been a thing for a couple of years

now. And Microsoft was really on the ball with that, making sure that that got patched as soon as they were aware of it and the patch was available before any of the exploits from the kit were publicly available. And still to this day, we see the problems, but would it have reduced the attack surface?

Kurt:

Yeah.

Tony:

Oh, sorry.

Kurt:

I said they even pushed XP patches when they stopped even patching XP at that point. I mean, Microsoft did a great job, making sure that they could help people with the issues.

Tony:

I mean, on the one hand, it would've been nice to get the advanced warning. And on the other hand, as we've mentioned, well, Log4j2, it's an open-source project. And my understanding is that there are like three people running it. They don't get any funding, and it was just a project for them. So I'd have to believe that they're doing the best that they can right now.

Kurt:

Tony, do you think there was a better... Like what Heather asked, do you think there is a better way than just publicly announcing the issue? I mean, since it's open-source, I understand probably why that's the route that was taken on it, but–

Josh:

... it's also how many people would you tell? Go and try to find every place where this... You don't know.

Kurt:

I mean, it forces people to fix it, which I definitely think is good considering its open-source and used everywhere but–

Tony:

... I mean, on the other hand, sometimes it isn't always a direct job between we've made these code changes to this is the vulnerable function. I don't know if you guys remember the speculative execution thing in Ghost from a while back; a lot of people were asking why certain versions of the Linux kernel or certain updates to macOS, why they were disabling speculative execution? And then why are these Intel execs dumping all of their stocks? And then a week later, we find out about Ghost, and we find out about all these problems with how speculative execution works and... So the typical method of disclosing vulnerabilities is that you do the responsible disclosure thing. Either you talk to Google Project Zero, or you talk to the vendor, and you give them X number of days, and hopefully, they have a patch by then. And then you wait until the patch is public to kind of discuss your findings. It might have reduced the attack surface a little bit, but this is the world that we live in, and it's not illegal to drop zero days on GitHub yet. So this is what we had to deal with. I say, yeah because–

Kurt:

... I appreciate the yet–

Tony:

There were cases where with ProxyShell, where people were trying to put proof of concepts for the exchange vulnerabilities, and Microsoft was taking them down almost immediately because they own GitHub. That's what I was just saying, too there is I don't want it to be illegal; it's security research–

Kurt:

... no, that–

Tony:

... that's what we have to do.

Kurt:

That's a whole different ramble. That goes down a rabbit hole of third parties having the ability to complete and monitor what's posted in there, but that's not the point of this.

Tony:

Yeah, that is a whole other can of worms.

Kurt:

Oh, yeah. Strong feelings on that one, but that totally could cause problems. So let's say something came out about Microsoft, Tony, and now, people are trying to get it out there on GitHub and anything. And they're justly removing it because they have every right to, I mean, there're other platforms to push it to, but either way, I won't derail us.

Heather:

What about the 2016 Black Hat was this vulnerability talked about then?

Tony:

Well, to some extent, it was. When we talk about those deserialization vulnerabilities, that's the research that was going on at the time. I got a slide back together that I'm putting together for a video, and I linked... Or, I set up a couple of different resources where JNDI was known to tax surface. So I got a couple of guys here. One of them was Chris Frohoff and the "ysoserial" research that he did and Oleksandr Mirosh, which was the Black Hat talk that you're referring to and then Bechler and the marshalsec tool and Java on Marshal Security. So those were like the big pieces of research that really opened up this whole can of worms into being able to use existing gadgets and Java libraries and the whole idea of using the LDAP attack surface to force applications to load class files. So yeah, this has been a known issue for a number of years now, probably, maybe even before the Black Hat talk. But nobody really knew it was a tax surface in Log4j2 because nobody really expected there to be the functionality to do lookups. I mean, it's just a logging library. You wouldn't really expect that to be there.

Heather:

Before we wrap up here today, Josh, I think you wanted to talk about something else that you're seeing.

Josh:

That's something that's come up with some of these scanners, where they patch the vulnerability, like the vaccine; people turning that into anti-worm to go and patch things. And I believe there are efforts out there using that same type of code to write worms to spread it, as well.

Kurt:

See, the only concern, though, I can... I think at the core of that; it's meant to be proactive and help people. Obviously, I don't think the intentions of that are malicious by any means. But if we go back to when Tony was talking about critical infrastructures and stuff and then tie it to what I was talking about, you probably have to update your Java and stuff to even maybe patch for Log4j. I mean, you take all that into consideration. Let's say a worm gets into your environment and starts patching stuff that needs to be running, but now it all starts crashing and failing. All because someone was attempting to patch your environment for you and help resolve the problem. But in return, they just made everything a living hell for you.

Josh:

Yeah, none of that is good all around. Although, in this case, like the worm ability of this isn't as bad as something like EternalBlue, where it's one exploit that can be scanned for and exploited the same way everywhere. At least, in this case, it depends on what application you're running, what the attack surface is, and how it's exploited.

Kurt:

Gotcha. I haven't read too far into how they're–

Josh:

... it'd be hard to write a worm that would affect everything. And that's the same problem with things like the vaccine thing with the scanner, where it's only going to work in some cases.

Tony:

What's really interesting is that there are parallels that can be drawn between this and Shellshock. There was a couple of different campaigns when Shellshock came around that were targeting network-attached storage devices. Where they would exploit Shellshock, get on the system, patch the vulnerability, set up a back door, and then start attacking another system and keep doing the same thing over and over again. So I just thought that it was pretty interesting; the suggestion about it being wormable. We definitely have seen that in the past.

Heather:

All right. Well, thank you guys for joining me today. I really appreciate you taking the time. That's all we have for now. We'll be keeping an eye out for updates as information about this vulnerability develops. Do be sure to check out our links. We've included a few resources for you to learn more. Until next time. Stay safe.