Heather:

Welcome to the hurricane labs podcast. I'm Heather. And this week we're doing a special double release to follow up on the Log4Shell vulnerability. This follow-up is a two-part series. So be sure to check out SOC Talk Log4Shell when we release it on Friday this week. For today, we're going to be hearing from two of our pentesters, Collin and Dennis. They're going to be sharing a bit about what they've seen with regards to this vulnerability and what security teams should learn from this red team perspective. So Collin, Dennis, thank you for joining me. Now that Log4Shell has been around for a few weeks. What's been your experience, have you had any pentests come up where you've used this?

Dennis:

Have you done anything with it? Collin?

Collin:

Yeah. So I went through some of the POCs and wanted to test it out to see how it works. And it's pretty trivial to set up. I think the blue team people probably have it a lot harder than we do in terms of responding to this because exploiting is a much easier task.

Dennis:

Yeah. The exploit's easy. And it's also... so it's in a language, it's an injection that's in a language and the language is more cursive. You can insert some syntax that makes it like a format string or a template where you can put this variable in there. But if the variable contains more of that language, then it recurses deeper and does it again. So it tries to parse the outer dollar sign, curly brackets, and then as it does that, if it runs into another dollar sign curly brackets, it goes deeper. Which means for us as pentesters, there's going to be a ton of bypasses. You have a lot of stuff to play with in order to attempt to bypass web application firewalls and other kinds of things. Unless if whatever defense system you have has a parser for this, then I don't think it will be easily stopped. Except for with the patch itself, it's going to be everywhere and it's going to be in the weirdest spots. So even if I'm testing something that's ASP or that's not Java that is not supposed to have Java anything thing to do with it. Log4j is in so many weird places that there might be some outside logger somewhere else that somehow gets information at some point or another, that ends up getting logged and causes the vulnerability.

Heather:

Where are some unexpected places that you have been seeing or hearing of Log4Shell being an issue?

Dennis:

I've seen on Twitter, someone had showed a picture of a gas pump that they used their credit card on and their credit card, they got it with a Log4j payload in the credit card and it crashed the gas pump.

Collin:

Oh my gosh.

Dennis:

Yeah. Because like how do you update a gas pump? They're going to be slow to update. So we forget that it's going to be in the weirdest places. I heard a story of a casino that got ransomwared because they broke into the fish tank.

Heather:

Yeah. I remember hearing about that. So smart appliances are an issue.

Dennis:

That's the idea. If your fish tank is running Java or fancy coffee machine that runs off Java, then you could get ransomed because of Log4j and you forgot to update your fish tank. So that is one of the biggest scary aspects of this is, well the big storm seems to be over now. Later, we're going to find out just how bad it is. And then later after that, we'll find out it's worse. I suspect.

Collin:

Yeah. It's going to be a handful of people who haven't got the bulletin and their fish tank gets popped despite everything else being up to date.

Dennis:

Like is there medical devices running log, running Java? I bet there are. So that's the most scary aspect is that it could be anywhere, not to be fear-mongering but that's part of the scary part. One of the big takeaways is how ubiquitous this is. And that Java, I feel like is one of those things that like steeps into everywhere that even if you're a C++ developer or whatever it's going to be everywhere, is one of the big takeaways. You probably know more about that than I do, because you've done a lot more with Java than I have.

Collin:

Yeah. Like I think that's the big thing because first of all, Java runs on like 3 billion devices or so they say. And Log4j, I guess it helps to have some background. It's like a logging framework. So to kind of help facilitate writing error messages and like recording information, debug information or errors. So a lot of tools use this and it's a very basic vulnerability in that all you need to do to exploit it is to get your string or get a piece of text, to get sent to one of the logging calls. So a lot of times a common example is web servers will be logging user agents string or user-supplied information, or even like a username. So and so this IP tried to log in as this user. Well, you can put the specially crafted string that has a JNDI reference, which is an enabled by default feature of Log4j, or at least it was. And it allows... basically allows you to retrieve information and Java objects from other sources, which is kind of an issue because it's on by default and it's not... if you're not otherwise aware of it and you're just using Log4j you're immediately vulnerable to this vulnerability without even knowing it. So that's kind of another big issue of this that I think is finally changing and that they've been patching out.

Heather:

What else do you find concerning about this vulnerability?

Dennis:

From a pentester perspective is the potential to miss it because of egress filtering. So in general, in order to attack this thing or to detect it, you send this string that causes a LDAP or an RMI query outbound. So let's say you have a web server that's vulnerable. I send it the evil payload and then the server itself doesn't tell me, doesn't respond to my web request saying, "oh wait hey, I'm vulnerable." It makes a DNS request and an LDAP or RMI request in order to get information out of band. So this is essentially has nothing to do with HTTP. It's nothing to do with the protocol I'm sending, the web server being

vulnerable, sends something else out. And if that is filtered by the web server, by some tool, or something like that on the network, then, as far as I know, it's not vulnerable. I never saw a call back to my DNS server. I never saw a call back to my LDAP server, as a result, I can't detect if it's vulnerable and it hides. And so it might hide for longer until someone finds a bypass that works for your specific situation. Maybe they find a way of hosting the vulnerability on GitHub so that your LDAP query retrieves a Java object from GitHub's domain, which is white listed inside of your egress filtering or something along those lines.

Collin:

Yeah. Those are good points. And it's kind of unfortunate that from the reporting side because this has a CVSS rating of 10. That's like top marks, you've aced the... you maxed out your vulnerability here, but our ability to confirm the existence, the actual exploitability of it when egress filtering is in place, like Dennis said, kind of makes it a challenge because the best you can put on a report is that we think it might be vulnerable based on version numbers or some other behavior or worse, just not have it appear at all. So it's very high risk to have, and very easy for it to just not be visible to us on the testing side.

Dennis:

So if you do get a pentest with us or with anyone else, it's good to make sure that Burp Collaborator or any other callback server that the pentesters are using is whitelisted. Burp collaborator, you don't have to worry that Burp Collaborator I can't use that to cause code execution, in this case, I'll just get a DNS in a connection request, but Burp Collaborator, won't answer with a Java object and get code execution. I think you can trust Burp in that way. And the way they've designed Burp Collaborator is to make it anonymous so that they don't know who's talking to it using some neat cryptography tricks. But if the Burp extension that I mentioned earlier, all of those payloads call out to Burp Collaborator and one of the neat things about this vulnerability is you can tell whether it is this vulnerability or a different reason that the callback is happening just by embedding a second payload inside of the URL. So for you pentesters out there and bug bounty hunters, don't just put the JNDI RMI, or JNDI LDAP request that you see on Twitter inside of your payloads, check to see a DNS request... get a DNS call back and say, that's the vulnerability because it might be a false positive. There's a lot of things that anytime it sees anything look like a URL or a domain name, there's various things that just reach out to find out what the IP address is, which will cause the DNS request in a false positive. You can embed a second variable in there. So instead of saying, call out to hurricanelabs.com, you say, call out to Hurricane Labs dot dollar sign, curly bracket, lower colon F, lower colon whatever, in order to add a second variable in there. And if the DNS request has a lower case F or a lower case whatever, whatever payload you used, then that means it was evaluated in its Log4j doing the out valuation. And you can be confident that it's actually Log4j and not some other system just looking up DNS stuff. So in that sense, Log4j is very kind to us pentesters, that if we get a callback, we can verify that the callback is a result of Log4j parsing. But if we don't get a callback, there's no reason for us to believe that there's a vulnerability in trying to guess what your egress filtering is. And for us to register multiple domains in order to... Domains cost money to register, and it's not easy just to start registering them in bulk in order to find what your egress filtering allows. So it's good just to... the outbound stuff that the pentester asks be whitelisted, it's good to whitelist that, just to make sure you're catching it. At least for the duration of a pentest.

Collin:

Yeah. And that's a good point about the URL scanning, where you might be inclined to send off one of those JNDI strings with a URL in it, and you see a DNS query back when in fact it's just an automated...

Kind of like how Google or I think other services like Discord will kind of autoscan them, but yeah, those showing up as false positives. Similarly, in that vein, you can apply WAF bypass attacks in there, kind of like what you were talking about, where some web servers might... some mitigation techniques might involve just looking for dollar sign, brace JNDI do some basic regex for that, but it's worth knowing that that's not an effective mitigation strategy. And you see this with JavaScript and other things where you can insert sub-expressions into it that get evaluated, like lower colon, whatever, which makes the character lower case. So you can kind of really mangle the input and on the server, when looking at the string itself, doesn't look malicious and it doesn't match the traditional pattern, but when it gets evaluated on the server, it will be kind of resolved down and execute actually malicious code.

Dennis:

Yeah. As a pentester, I'm sure Collins' thought of this too. I've run... I'm sure you've run into this where we'll find a vulnerability, usually happens with cross-site scripting, which is a similar vein, like you was saying where JavaScript is very difficult to parse because of how expressive it is. So it's very difficult to detect whether something is JavaScript. I can make JavaScript look like something that is not JavaScript at all. Many times I've had fine cross-site scripting in some system that could not easily be patched. The vulnerability itself couldn't easily be addressed because it's not the client's system, or it's not the client's code. And so they want to put a bandaid on it for now to at least move forward and offer them some protection. And so a WAF is a great choice for that except WAF's don't ever work out of the box. With cross-site scripting, pretty much every single time that they've just added a WAF I've immediately bypassed the WAF. On occasion, it's worked. Usually, it takes some fiddling in order to get the WAF to be effective at all. And even if you fiddle with it a bunch, quite often there's situations where the WAF has to be more restrictive than the website will allow. And normal traffic will end up getting blocked by the WAF depending on how the injection works. The JNDI vulnerability, this Log4j isn't nearly as bad as JavaScript and cross-site scripting. It is very difficult. I don't think you can exploit this without the dollar sign curly bracket. So unless the server allows multiple layers of encoding or something like that, then you will see dollar sign curly brackets, which I should say multiple layers of encoding is not uncommon at all. That's one of the best ways to get cross-site scripting passed to WAF but even still adding in those sub-variables, whatever you want to call them inside of the URL and some languages you're not allowed to make verbs in the language out of variables, but you're allowed to make parameters to those verbs out of parameters or the language. Log4j is not like that. It allows you to use a variable inside of JNDI. So if you're searching for JNDI you're going to have a bad time. You can instead make JNDI all variables like dollar sign, curly bracket, lower colon J curly bracket, and then dollar sign, curly bracket, lower colon in curly bracket, the whole way through, or instead of lower, you can use upper and then parse that into a lower, or you can take a known environment variable and reference a specific character out of that environment variable to be used for the JNDI string. There's all kinds of different tricks that you can use in that way. And so, unless, like I said, unless you actually have a parser that understands this, it'll be very easy for us attackers to bypass regular expressions or something like that.

Collin:

Nicely explained.

Dennis:

Yeah, that is one of the things that... So as an attacker, when there's a vulnerability, so I'm not really an attacker, I'm a pentester. It's my job to find the vulnerability. I try to exploit it in order to show the client the severity of it because maybe somehow I did get into your coffee machine and I can see what coffees

you're brewing. But if it's not connected to the domain, if it's not connected to the network in some way, that kind of a thing, maybe I don't actually get anything good out of it. So in that way any anytime as a pentester, I want to find every vulnerability that exists on your computer. I'm not going to, this Log4j the vulnerability existed for what was it like two or three years before it was discovered?

Collin:

I think the first feature might have been introduced in 2013 even. And then there was like a Black Hat talk on JNDI and LDAP lookups.

Dennis:

Yeah.

Collin:

In like 2016.

Dennis:

Yeah.

Collin:

It's crazy how long it has been around.

Dennis:

But no one discovered it, even though it's everywhere. No one discovered it until this year. So pentesters, aren't going to catch everything. But if a pentester's good, he should be catching what other good attackers would also catch. And so that's the merit of pentesting is the things that are known to other attackers. Your helpful pentest attacker can find for you. And that's why you want to make it easy for the pentester. You make it easy on the pentester, you don't egress filter, Burp Collaborator during your pentest. And then you make it harder on the real attacker and do all kinds of egress filtering for them. Because if I can't get anything and you make it easy on me, then the attacker who has it worse is going to be in more trouble.

Collin:

Yeah. It's important to cooperate.

Heather:

What resources can you suggest for identifying if or where you have a Log4Shell problem?

Dennis:

With this vulnerability being as easy as it is, just log the payload yourself on purpose and see if it makes a DNS request. You may as well. So there's a bunch of different tools that will find this vulnerability. One of them that I've seen is there's a Burp Suite extender, which is like a plugin for Burp Suite called Log4Shell-Everywhere. And it's based off of... James Kettle had Collaborator-Everywhere, which was just like sending a URL in everything that he could find in hopes of finding some kind of locking mechanism that reached out to URLs whenever it found it. And he found all kinds of really weird vulnerabilities with

that and gave some talks about it. So that inspired this extension called Log4Shell-Everywhere, which makes every single one of your requests that go through this Burp proxy. It shoves a Log4j payload everywhere inside of an extra URL parameter in refer, in via header, in X-Origin-URL, in destination in forwarded in X-API-Version in X-Arbitrary in X-Real-ID or IP, sorry. So all kinds of stuff that a lot of these are like proxy things and weird HTTP stuff that you don't normally see, but might get logged somewhere. Especially if the request is going through a proxy or a WAF or something like that. There's a lot of really sneaky, hidden spots where this vulnerability might exist, that you may not catch onto.

Collin:

And there's also an NSE script that someone has released where you can give it different payloads and run that against obviously an Nmap. So in cases like that, that might allow you to test a lot of hosts all at once. But I don't think it's as comprehensive, at least for websites as the Burp one.

Dennis:

Yeah. NSE scripts are awesome. You can also check out ZAP. ZAP has... I know they have at least a couple extensions or plugins that do that act like Burp. ZAP has gotten really cool. They do a lot of automated stuff, so you can script ZAP to do stuff. So it's also a great means of testing for this. I suspect I'm not sure. I suspect they have something like Log4Shell-Everywhere in ZAP, but if not, it's comparable, I believe.

Collin:

Yeah. It has to exist. And if it doesn't, it will very soon.

Dennis:

Yeah. Maybe I'll add it myself. ZAP is free and that's one of the... it's free and open-source, whereas to use Burp extension... No, I think you can use some extensions for free on Burp, but not all of them.

Collin:

Yeah.

Heather:

What I saw, I'm pretty sure the one you're talking about though, you have to have the professional. I'm pretty sure.

Dennis:

That makes sense cause we need the collaborator, the collaborator with Burp Professional only. You might be able to... if Log4Shell-Everywhere is open source, you can probably modify. It looks pretty simple they're just jamming a string in everywhere. You could change the string to your own DNS server if you have one, or I think there's a couple public Burp Collaborator type things that aren't Burp Collaborator that you can sneak it in there, depending on of course, whether the license for Log4Shell-Everywhere.

Collin:

Yeah. There were two that I was kind of collecting, haven't tested myself, but the two that I was looking at where log DNS is like the self-hosted DNS server for logging that type of stuff and like Canarytokens. I

think it's an online service that it generates a token, a unique DNS server for you. If anyone visits or if anyone requests it, you can see who requested it. So those can be useful for testing. If you aren't able to use Burp Collaborator or depending on your circumstances.

Dennis:

Additionally, I haven't checked it out, but this is also a good web extensions or web plugins. It would be easy to design a web plugin that does the exact same thing. So I'm sure it exists somewhere. And that might be the easiest solution at all. Cause you don't have to download anything separate, you don't need ZAP or something like that. You just install on Firefox's or Chrome's page. The web extension, it just does its thing. It'll probably have to use a public or a private DNS. You might have to set up a DNS server or point it at your own DNS server in order to get the results because it is an outbound request. But otherwise, that makes sense.

Collin:

I wonder if an extension like that would get past the Firefox or the Mozilla extension store guidelines.

Dennis:

Yeah. It's kind of...

Collin:

Safe and secure.

Dennis:

... It would be kind of weird. I think it probably would if it was obviously for catching it, but even still you don't need it to be signed in order to add it to Firefox as in debug mode or in Chrome as debug mode.

Collin:

Mm-hmm (affirmative).

Dennis:

So you can add it even if Firefox doesn't allow you to. It just won't... You have to explicitly add it every time you restart the browser though. If it's not signed by Firefox or Chrome.

Collin:

Yeah. That would be amusing to submit that to the store though.

Dennis:

Maybe I'll submit one and see what happens.

Collin:

You should. I'll give it a star.

Dennis:

Yeah. Collin is the only person to give a star so far. I think to my Firefox plugin that I wrote for catching cross-site scripting.

Collin:

It's called Eval Villain, go check it out for all your JavaScript testing needs.

Heather:

Eval Villain just got added to ZAP too. Right? I'll include links to your blog and your post on ZAP so that our listeners can check them out. As for Log4Shell. What's the big takeaway that people should get after our talk today?

Collin:

So you can't trust the output from Log4j or the Java end on this side, I think is what it kind of boils down to. But whatever's handling those requests like a proxy or a non-Java web server. You'll be able to see those strings. And there are some good documents out there for basically grepping through your log files to find where JNDI and these different attack vectors might be used.

Dennis:

Yeah, like you said it's kind of weird. You will and you won't see this in logs. If you don't see it in a log, that's the scary part, because that means you're vulnerable. It means that it was replaced with an actual Java object and someone could be running in code. If you do see it in the log, it means that whatever logged it itself is not vulnerable, but there might be something further down or further up that is logging that string and vulnerable. So those kinds of things, I think the blue team kind of guys are probably better at speaking at, but in general, I think for the takeaway from the pentesters is you never know... As a pentester, I'm still surprised where payloads end up landing in weird ways and causing execution in strange places. And that's going to be the big thing with Log4j is it reminds me of a virus or of COVID that if it gets anywhere near... If Log4j's vulnerability gets anywhere near a Java application, it'll be infected and you need to patch, you need to get vaccinated for your servers. And you never know what servers are sitting around unvaccinated, so to speak. That's the thing that I would be worried about. If I was someone with servers, what is using Java?

Collin:

Also, if you use Log4j if you're using JNDI how you're using it, and if there are any places where a user or a piece of data might be able to get into there. It definitely comes down to a lot of testing. I think a lot of the responsibilities also on the Java devs and those build systems to make sure that you're using the patch libraries, or if you can't update, just go in. As a red teamer, I can't really speak much, I think to the blue side of it. But it's often as simple as removing the classes for that JNDI lookup. If it wants to use it, it can't because it's not even available in the library. So that's one advantage or one upside to using Java. You can yank out classes pretty easily.

Heather:

All right. Well, that's all for now. Remember to stay tuned for part two of our Log4Shell follow up series, I'll be chatting with some of our SOC team about what they've been seeing these past few weeks and what it means for your security. Until next time, stay safe.