Anatomy of a Log4j Exploit

Description: This week we start off by looking at how the U.S. Pentagon is dealing with Log4j and how the U.S. administration at the White House wants to improve the security of open source software. This being the third Tuesday of the month, we'll look back at last week's decidedly mixed-blessing Patch Tuesday, the good and the unfortunate. We'll check out a very serious new remotely exploitable problem which affects many popular routers, and provide a shortcut of the week to immediately check your own routers. We'll cover a new and very welcome access control standard being introduced by the W3C which Chrome is already in the process of adopting.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-854.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-854-lq.mp3

We'll wrap up the top portion of the podcast with yet another set of very serious WordPress add-on blunders. We'll share a bit of listener feedback, including answering the very popular questions about refilling empty SodaStream tanks. And after a brief SpinRite progress update we're going to take a close look inside the operation of an actual Iranian Log4j exploit kit.

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots to talk about, one week into the January Patch Tuesday update. Steve says it's decidedly a mixed blessing. There's a new router exploit that's going to get you to want to reboot your router, and then a step-by-step examination of a real-world zero-day Log4j exploit. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 854, recorded Tuesday, January 18th, 2022: Anatomy of a Log4j Exploit.

It's time for Security Now!, the show where we cover your safety, security, privacy online with the man in charge, Steve Gibson. Hello, Steve.

Steve Gibson: For the 854th time.

Leo: Wow.

Steve: Yo, we're getting the hang of this, Leo. It's like, okay, just increment the counter, add seven to the day of the month unless it wraps, and off we go.

Leo: In fact, I can almost always go, oh, there'll be Microsoft Patch Tuesday exploits.

Steve: That's right.

Leo: There'll be something wrong here, some ransomware there. It writes itself, Steve.

Steve: You're right. What am I doing here? I can sure...

Leo: No, no, no, please, you don't get out of this. No way.

Steve: So we're going to start off by looking at how, well, I should say this is Episode 854 for the 18th of January, "The Anatomy of a Log4j Exploit." We're going to start off by looking at how the U.S. Pentagon is dealing with Log4j, and how the U.S. administration at the White House wants to improve the security of open source software. This being, as you said, Leo, the third Tuesday of the month, we'll be looking back at last week's once again unfortunately decidedly mixed blessing Patch Tuesday, the good and the bad. We'll then look at a very serious new remotely exploitable problem which affects millions of popular routers, and provide a shortcut of the week to immediately check everyone's own router to see whether they might be vulnerable to this. And if so, just unplug.

Leo: Oh, man.

Steve: Oh, it's bad. And then cover a new and very welcome access control standard being introduced by the W3C which Chrome is already in the process of adopting. We'll talk about that. This is really a nice piece of technology. We'll wrap up the top portion of the podcast with yet another set of very serious WordPress add-on blunders. I think it was 84,000 sites affected now?

Leo: Yikes. Yikes.

Steve: It's like, oh, goodness. We're going to share a little bit of listener feedback, including answering the very popular questions about how to refill empty SodaStream tanks.

Leo: Turns out the topic of the week last week.

Steve: It was. You and I just briefly mentioned it in passing. I got swamped in Twitter. It's like, what? You can do that? How do you do that? So we'll talk about that. And then after a brief SpinRite progress update, we'll take a look at the inside of the operation of an actual Iranian Log4j exploit kit.

Leo: Oh, wow. Oh, wow.

Steve: So, yeah, I think a fun podcast for our listeners.

Leo: All right-y.

Steve: And of course an engaging Picture of the Week.

Leo: Yes, I love it. You know, this is the geekiest show we do. I just love the level of, well, you'll see when you see the Picture of the Week. I just love it. I just love it. All right. I have a Picture of the Week that made me laugh out loud. I love this.

Steve: So, yes, for those who are not looking at the screen or the show notes, this begins with the inspiration from the flag of Norway, which is set against a red field. And basically it has what looks like, well, when you see it, sort of like two wires crossing perpendicularly. And they're insulated by some whiteness. Anyway, somebody realized, well, we could do some other flags. For example, you could have, instead of Norway, you could have ANDWAY, where those who are familiar with logic diagrams would replace the crossed wires with an AND gate. Of course then that would lead to NANDWAY, XORWAY, XNORWAY if you invert the output of the XORWAY, and of course NOTWAY. And having looked at those, I realized, well, it's really too bad that we began with NORWAY as the...

Leo: Right?

Steve: Because, I mean, it's an OR gate. So, yeah. But no. That was the inspiration, so we're leaving that one alone.

Leo: And by the way, that AND gate is the inspiration for the TWiT logo.

Steve: Ah, the TWiT logo, yes.

Leo: Yeah, yeah.

Steve: That's very cool.

Leo: There you go. All right.

Steve: Okay. So Hack the Pentagon with Log4j. At the end of last year, 2021, the Pentagon pivoted its ongoing Hack the Pentagon bug bounty program, which we've talked about years ago, I think it was 2016 that they launched it. And that's something that's being managed through HackerOne, through the HackerOne program, as opposed to trying to do it in-house, which I think was a good idea. They pivoted, not surprisingly, to track down Log4j vulnerabilities on what amounted to potentially thousands of public-facing military websites.

This was the first time that the U.S. Department of Defense had marshaled the ethical hacker community to tackle an emerging digital crisis in essentially real time. So

normally it's just sort of like, oh, well, if you find something, and you're prequalified, remember that they - I think they have 50, something like 50 vetted cybersecurity researchers. They're not saying anybody anywhere come hack us. That'll probably get you in trouble. You need to be preapproved.

So just after days from the time that the public was made aware of the Log4j problem, the branch of the DoD known as the Defense Digital Service, or DDS, in connection with HackerOne, who as I said manages the department's bug bounty program, had broadened the scope of the ongoing competition for testing their own systems and software. Katie Olson, the director of the DDS, told The Record that: "It was a really quick effort, and a really elegant solution, to use a contract that we already had in place with the crowdsource research community to very quickly do a scan of what might be affected within the DoD."

Okay. So as a result, the roughly 50, as I said, previously vetted cybersecurity researchers who were already participating in the existing hunt were given the additional assignment to scour all .mil websites and report any potential weaknesses or exploits caused by the Log4j vulnerability. This on-the-fly change, that is, in their targeting, coincided with the decision we talked about last week by the U.S. Department of Homeland Security, whose own bug bounty program was just in the process of being launched, to similarly broaden the scope of its own bug search.

Major tech companies and federal officials also have scrambled to grasp the full extent of the Log4j flaw, warning that potentially, as we know, hundreds of millions of devices around the globe could be compromised. CISA last month issued an emergency directive requiring all civilian federal agencies to mitigate the threat. As I've often joked, no Christmas until you do, even though Christmas did happen on schedule, though top agency officials on Monday, last Monday, repeated that they have not seen a malicious actor use the vulnerability to breach federal departments and agencies. At the same time, there's the expectation that, if that had been done, it would be definitely kept on the down-low.

During a conference call with tech reporters, Eric Goldstein, who is CISA's executive assistant director for cybersecurity, stated that the effort had already, that is, the government's effort, had uncovered 17 previously unidentified assets that were vulnerable to Log4j, all, Eric said, which were remediated before any intrusion could occur. He added that: "It demonstrated the extraordinary power crowdsourcing brings to the research community to help not only the U.S. government but the broader nation to find vulnerabilities before adversaries can abuse them."

So anyway, although the Pentagon was already using an ecosystem of passive scanning software and technology to continuously monitor its assets, as we know, Log4j differs from previous cyber incidents by not centering around specific types of hardware or software, like VPNs or firewalls. The trouble was, at the time of its initial disclosure, there was no mature automated solution available to track down, locate and verify exploitable vulnerabilities. A guy named Lance Cleghorn, who's a digital services expert at the DDS, told The Record: "That's where the crowd really comes in to save the day. They can not only tell you, 'Hey, I actually went and found this, and it's vulnerable for sure,' they're able to also say, 'Here's the evidence.' And 'It's exploitable, and that's a problem.'"

So at first blush public-facing military websites may not seem like an attractive target for hackers. Again, you do risk bringing down the wrath of the U. S. government and the DoD. However, there's long been a concern within the DoD that a sophisticated threat actor could use a previously unknown vulnerability to penetrate its network - yeah, like anybody on the 'Net should be worried - and to gain a foothold in the department systems. And there is that massive Nonclassified Internet Protocol Router Network known as NIPRnet, literally stands for Nonclassified Internet Protocol Router Network.

HackerOne's CISO, and who also calls himself the Chief Hacking Officer, Chris Evans said that once the bug bounties were expanded to explicitly include Log4j, hackers "responded immediately and competently, with numerous valid reports pouring in within the first few hours of that explicit expansion." The DDS paid competitors $500 per discovered vulnerability and an additional $500 if proof of exploitability was also provided. And I don't know, Leo, you know, given the amount of money that the U.S. has, every time I look at those numbers that...

Leo: Seems so cheap; doesn't it?

Steve: It does. It's like, if you're serious about getting these problems rooted out, you know, a couple grand at least; you know? Really.

Leo: I agree.

Steve: Wow. However, neither Katie Olson nor Lance Cleghorn, both at the DDS, were willing to disclose how many vulnerabilities had been found during the retooled bug bounty. Lance did say: "We've paid out a chunk." Okay.

Leo: A whole $700.

Steve: Exactly.

Leo: A lot of money.

Steve: Exactly. And Katie hopes that even more U.S. government agencies will move to establish their own bug bounty programs. You know, it does require some focus. But one would hope that they already have - they've already got to have IT; right? And they've got to have security people. If they don't, they've got bigger problems. So how tough can it be to establish a relationship with HackerOne and just, you know, copy what the DHS and the DoD have done. It's been done; right? There's already agreements and contracts in place. Just change the heading on the top of the page and then get the other agencies going. I can't understand why they wouldn't have done that already.

Speaking of the U.S. government, and specifically the White House, last Thursday the 13th, the Biden administration convened what they called the Open Source Software Security Summit, having the stated goal of getting public and private sector organizations to rally their efforts and resources with the aim of securing open source software and its supply chain. Okay. Good goal. Although not only about Log4j, Log4j was the clear catalyst behind the summit.

In the public sector, the list of participants pretty much was the Who's Who, including the Deputy National Security Advisor for Cyber and Emerging Technology, that's a department, that was Anne Neuberger; National Cyber Director Chris Inglis; officials from the Office of the National Cyber Director; Office of Science and Technology Policy; the Department of Defense; the Department of Commerce; the Department of Energy; the Department of Homeland Security; the Cybersecurity and Infrastructure Security Agency, of course, the CISA; the NIST; and the NSF. The private sector was well represented by Akamai; Amazon; Apache; Apple; Cloudflare; Facebook/Meta; GitHub; Google; IBM; the

Linux Foundation; the Open Source Security Foundation - the Open Source Security Foundation, I didn't know there was one, good; Microsoft; Oracle; Red Hat; and VMware.

The participants focused their attention onto three topics: First, preventing security defects and vulnerabilities in open source software. Yes, good, let's do that. Improving the process for finding security flaws and fixing them. And, third, shrinking the time needed to deliver and deploy fixes. All worthy goals.

The White House's after action report wrote: "Most major software packages include open source software, including software used by the national security community. Open source software brings unique value and has unique security challenges because of its breadth of use and the number of volunteers responsible for its ongoing security maintenance." So this sounds a little bit like what the FTC, we talked about last week, what they said, you know, they did appreciate the particular challenges it represented because it was not commercial. It was all free and just done by random people.

During the summit Google proposed the creation of a new organization that would act as a marketplace for open source maintenance that would match volunteers from participating companies with critical projects that need the most support. Kent Walker, Google's President of Global Affairs and Chief Legal Officer both for Google and Alphabet, he was quoted saying: "For too long, the software community has taken comfort in the assumption that open source software is generally secure due to its transparency and the assumption that many eyes were watching to detect and resolve problems. But in fact, while some projects do have many eyes on them, others have few or none at all. Growing reliance on open software means that it's time for industry and government to come together to establish baseline standards for security, maintenance, provenance, and testing."

Leo: Just support it with money, for Christ's sake, at least.

Steve: Exactly.

Leo: You know, that's the problem. They use this stuff for free, and then they go, well, you see.... Support it.

Steve: Yeah, it's broken.

Leo: It's broken. Support it.

Steve: "...to ensure national infrastructure," he said, "and other important systems can rely on open source projects."

Leo: At least he's not saying only use proprietary.

Steve: No.

Leo: But it is Google, after all.

Steve: Right, yes. "These standards should be developed through a collaborative process, with an emphasis on frequent updates, continuous testing, and verified integrity." Yes. Wouldn't that be nice. If only, as you said, Leo, we had the money. So, yeah, this is nice to see. But for me, at least, I have no idea how we would get from where we are today to there.

The end of the White House's report suggested that the government's purchasing power could be, and would be [yawn], used to bring about that change. What was written said: "President Biden has made software security a national priority." Okay. I'm going to just bite my tongue. "His Executive Order on Cybersecurity requires that only companies that use secure software development lifecycle practices and meet specific federal security guidance will be able to sell to the federal government for the first time," okay...

Leo: Wow.

Steve: "...leveraging the purchasing power of the Federal government to drive improvements in the software supply chain, improvements that companies and governments around the world will benefit from." So if we had not been paying attention to the way things never seem to get done in Washington, that buying power statement might be encouraging. But it's just more bureaucracy. We know that our present system is far from perfect. It's a constant and necessary theme of this podcast. But efforts like Google's Project Zero, Trend Micro's Zero-Day Initiative, HackerOne's bug bounty management, the Pwn2Own competitions, the annual Black Hat and Defcon conferences, everyone contributing to Chromium, the academic research, and the occasional crowdsourced funding of intensive security audits of mission critical packages, these are all existing, proven, and highly effective solutions within their own realms which have all emerged organically. They have thrived year after year because they have been effective, and they've made sense.

If the U.S. government wants to help, its time would be better spent, I think, exactly as you said, Leo, in peeling off some taxpayer money. And not too much.

Leo: No, it wouldn't take much.

Steve: Since you don't want to wreck the status quo, you know, you don't want to wreck all this by just throwing money at it and having it all go sideways. And put some additional funding behind these existing initiatives that are limited in what they can do due to lack of support for personnel or the size of bug bounty motivations.

Back in 1965, the U.S. Congress created an independent agency known as the National Endowment for the Arts. It offers support and funding for projects exhibiting artistic excellence. Artists write proposals and apply for grants to receive funding. I'm all for change and for improving what we're doing. But we're also largely doing the right things now. Doing more of what we're already doing seems like a nearer term solution that could be implemented today with a greater guarantee of results. A National Endowment for the Improvement of Software Quality, if properly administered, might be a worthy consideration.

Leo: Good.

Steve: So, yeah. Maybe. Think about it.

Leo: Maybe, yeah, yeah. There's just so many open source projects that end up as part of commercial software, but nobody's paying for them. They're maintained by one guy's volunteer.

Steve: Yup.

Leo: And, you know, that's kind of Log4j's story. And but they're used as infrastructure. These companies need to kind of pony up and support this, I think.

Steve: Yeah, yeah. That fabulous picture of the - it's not a house of cards. It's like a tower of crazy blocks with, you know, down at the very bottom one little toothpick-size thing that's like holding up the whole crazy mess and maintained thanklessly by some guy in Nebraska. You know, it's like, okay.

Okay. Speaking of software quality, last Tuesday was another of Microsoft's all-too-frequent mixed-blessing Patch Tuesdays. Okay. So first, here's the good news: A total of 97 vulnerabilities of varying severity were patched. And there were also an additional 29 vulnerabilities fixed in Microsoft's Edge browser. Of the 97 non-Edge vulnerabilities, nine were classified as Critical and the other 88 as Important. Overall, the patches cover Windows and Windows components, Edge, Exchange Server - no surprise there - Office and Office's components, SharePoint Server, .NET Framework, Microsoft Dynamics, some open source software even, Hyper-V, Defender, and the Remote Desktop Protocol (RDP).

Dustin Childs with Trend Micro's Zero Day Initiative said: "This is an unusually large update for January. Over the last few years, the average number of patches released in January is about half this volume. We'll see if this volume continues throughout the year. It's certainly a change from the smaller releases that ended 2021." Okay. So Microsoft patched 67 bugs last month in December. Now we're at 97. Okay.

Now, Microsoft classes a zero-day vulnerability differently than we do here. My feeling is that we need to reserve the term "zero-day," which has unfortunately taken hold as click bait, for a vulnerability which is first discovered when it is observed being used in the wild. The point is that patching that puppy which is currently being exploited is much more important than patching a problem that's only potentially exploitable and which has been reported responsibly and privately. So nobody knows about it except the group that can fix it. But Microsoft also classifies vulnerabilities that have been irresponsibly and publicly disclosed as zero-days. And also regardless of how bad they are. So like, you know, somebody discloses a vulnerability that has the pizza come out cold. Oh, that's a zero-day. Oh, okay.

Leo: That's a CVE of 10, baby. That's critical, man. Don't mess with that.

Steve: You know, but, okay, so that's what Microsoft wants to call them. I can see their point, since the race is then on to get the world patched before the publicly disclosed vulnerability can be weaponized and actively deployed. So I accept their definition in this case. And that also means that a total of six unexploited, but published,, zero-day vulnerabilities as they want to call them, were also fixed last week. So overall, the breakdown by type was 41 elevation of privilege vulnerabilities, and we know those are bad once you gain a foothold; 29 remote code execution vulnerabilities, obviously never

good; nine security feature bypass vulnerabilities; which could be anything; nine denial of service vulnerabilities, which mostly means it's easy to crash something; six information disclosure vulnerabilities, something leaks; and three spoofing vulnerabilities. Okay.

And separately, those six unexploited but published zero-days, as they call them, which were patched, were - and remember there was a mention of open source. There was the open source Curl remote code execution vulnerability. We'll look at that in a minute. It's not clear to me how that executes remote code. It's interestingly funky. But anyway, there's also a Libarchive remote code execution vulnerability, definitely looks like you could execute code with that one. Both of those are the two open source ones. Then we have Windows User Profile Service elevation of privilege; Windows certificate spoofing; Windows event tracing Discretionary Access Control List (DACL) denial of service, so that crashes something; and Windows Security Center API remote code execution.

Okay. Those first two, as I said, Curl and Libarchive, which are the only remote code execution problems among those six, had already been fixed by their maintainers, but the fixes had not yet been incorporated into Windows until last Tuesday. So last Tuesday's Patch Tuesday updated Windows' use of those, Windows' inclusion of those.

Now, reading the details of the Curl problem, it's unclear, as I said, how it could offer remote code execution. Here's what they said: When Curl version >= 7.20.0 and <= to the one that fixed it, 7.78.0, connects to an IMAP or POP - so this is Curl connecting to an IMAP or POP3 email server to retrieve data using STARTTLS to upgrade the connection's TLS security, the server can respond and send back multiple responses at once that Curl caches. Curl would then upgrade to TLS, but not flush the in-queue of cached responses, instead continue using and trusting the responses it got before the TLS handshake as if they were authenticated. Using this flaw, it allows a man-in-the-middle attacker to first inject the fake responses, then pass-through the TLS traffic from the legitimate server and trick Curl into sending back data to the user under the assumption that the attacker's injected data comes from the TLS-protected server.

Okay. So that's a cool hack; right? Somebody figured out that Curl is going to want to elevate its security to TLS, assuming that the IMAP or POP3 server in the hello handshake indicated that it supports STARTTLS, and so that would happen. So this is a very subtle and clever bug that comes about as a side effect of the STARTTLS kludge, which is really what it was. It was the original way of providing email encryption over the traditional SMTP, IMAP, and POP ports before they obtained their own dedicated TLS connection ports which they have now. And it's exactly the sort of bug that tends to creep into systems that were being pushed to do things they were not originally designed to do, such as on-the-fly switching an unencrypted connection to using encryption.

Okay. So that's the Curl bug. And as I said, someone claimed that it could be used for remote code execution, though didn't explain in this disclosure how that could be so. Just looked like you could get a bad guy, a man in the middle could sneak some stuff into an email client query that didn't actually come from the then authenticated server. But the Libarchive bug, affecting versions 3.4.1 through 3.5.1, is a use-after-free flaw in its "copy_string" function when called from either "do_uncompress_block" or "process_block." So that one might well be leveraged for remote code execution if a bad guy found some way to get the user or the system to use Libarchive to decompress a specially and maliciously formed archive. That one I believe.

In any event, patching should not be postponed since many of these already do have proof-of-concept exploits published. Remember they're Microsoft zero-days. And as we often observe, attacks never get worse, they only ever get better. Mostly though, compared to other things going on right now, this is certainly not a four-alarm fire. So not for that reason. Oh, and if you encountered some of last week's breathless "Oh my

god, patch now, Windows contains a wormable flaw" press coverage, the reason I didn't lead with it is that for Windows IIS server to be vulnerable to it, which is what that breathless press coverage was about, requires enabling an obscure and non-default registry key. Under CurrentControlSet\Services\HTTP\Parameters, someone needs to have enabled something called EnableTrailerSupport and set that to one.

We're all familiar with the way HTTP headers work, where they form metadata such as cookie information, an asset's creation timestamp, probably its lifetime before expiration, like how long that the client is allowed to cache it before explicitly checking to see if it's been updated and so on. Well, it turns out that it's also possible - who knew? - for additional headers, in this case called "Trailers," to be included after a chunked-style encoded query or response. Just a few weeks ago we clearly covered chunk-styled encoding because that came up in something we were talking about.

Okay. So you can have headers after the content. Okay, what? This really feels like the HTTP designers ran out of important work to do and sat around asking themselves, "What else can we add?" And that never ends well. So they invented a previously unappreciated need, suggesting that it might be that a client or a server would not be able to fully form its query or response headers until after the body of the query or the response had been formed. No one knows why that might be true; but, hey, it could happen. Remember that since the dawn of the web this had never actually apparently been a problem. But perhaps they got their important work finished early, so they decided to define a solution for this one, anyway.

So, yes, it turns out since HTTP/1.1, in addition to Headers, it's also possible to have Trailers. But as I said, Windows' IIS server does not have that feature turned on by default. And since no one actually uses Trailers, it's unclear why anyone would have ever turned it on. But okay, until last Tuesday, if someone had actually turned it on, then yes, IIS could theoretically be exploited by leveraging some mishandling in its non-default enabled support for HTTP's unused and unnecessary Trailers feature.

Now, as for that flaw being wormable, it seems to me that requires somewhere for the worm to go. And if no one else running IIS has that unused and unneeded and disabled feature enabled, that's going to be one lonely wannabe worm desperately trying to propagate. Which kind of reminds me of my adolescence.

Leo: Not reproduce, propagate.

Steve: Despite all of this, since we all agree that worms are bad, and since the attack complexity is quite low, this non-threat earned itself a CVSS of 9.8. So that alone must be what the other tech press saw and thought, "Oh my god, 9.8."

Leo: The pizza's going to be cold. It's going to be so cold.

Steve: It is going to be a very cold pizza. Now, actually, I've been told cold pizza can be quite tasty.

Leo: It is. It's an excellent breakfast treat.

Steve: Maybe that wasn't the best example to use. In any event, what was never a huge problem is no longer any problem. Well, it might be. That was Patch Tuesday's good news. Here's the other shoe.

As Threatpost headlined their coverage, "Microsoft Yanks Buggy Windows Server Updates." So maybe there's hope for that worm after all. Threatpost wrote: "Since their release on Patch Tuesday, the updates have been breaking Windows, causing spontaneous boot loops on Windows domain controller servers, breaking Hyper-V, and making the ReFS volume systems unavailable." Whoa. "Microsoft has yanked the Windows Server updates it issued on Patch Tuesday after admins found that the updates had critical bugs that broke those three things."

People who were quite frustrated were venting all over Twitter. I saw one posting asking the question, "Does Microsoft even test these things before releasing them?" There was actually a great deal of frustration. I heard about this directly from many of our listeners and Twitter followers. In addition, it's been confirmed that Tuesday's updates for Windows 10 desktop machines were also breaking L2TP VPN connections. They no longer worked.

BleepingComputer was tracking this saga day to day and blow by blow. On Thursday they reported that Microsoft had pulled the January Windows Server cumulative updates and were no longer accessible via Windows Update. But as of that afternoon, Microsoft had reportedly not also pulled the Windows 10 and Windows 11 cumulative updates that were breaking L2TP VPN connections, and that was confirmed. So it's unclear how that went.

This is all the mixed blessing of Windows Updates recently. We're pushed to install them immediately with breathless, though in this instance unwarranted warnings of the sky falling from a server worm. But installing these things through most of 2021 and continuing that trend into 2022 has resulted in the loss of mission critical functionality. So, damned if you do, damned if you don't. Actually, "don't" appears the increasingly attractive option given Microsoft's recent side effect-laden updates. Let somebody else go there first, see if they survive. And if so, then cautiously follow.

This is important, and everybody gets to participate. Okay. This is not good. Okay. So the bottom line is might be time once again to check for router firmware updates.

Leo: Gosh darn it.

Steve: I know. But you can go to grc.sc and this episode number: grc.sc/854.

Leo: 854, yeah.

Steve: And that will tell you whether you're okay or not.

Leo: Maybe. Is it set up?

Steve: It should have given you more than that already.

Leo: Let me go directly. There we go.

Steve: Ah, there you go, yup.

Leo: [Crosstalk] my port 20005.

Steve: Yes.

Leo: Okay. That's not a port I'm familiar with.

Steve: Not a port anybody's familiar with. Okay. So...

Leo: Uh-oh.

Steve: The security - yeah. You're stealth. Good. The security research firm SentinelOne has discovered that some common code licensed by a number of prominent router manufacturers contains a highly critical, remotely exploitable flaw. Among the routers known to be affected are those by Netgear, TP-Link, Tenda, Edimax, D-Link and Western Digital.

Leo: Holy [indiscernible].

Steve: I know. So here's what we know. They, or rather he at SentinelOne, his name is Max, discovered a high-severity flaw in what's known as the, well, KCodes is the company, KCodes NetUSB kernel module used by that large number of network device vendors and affecting millions of end-user router devices. This allows attackers to remotely exploit the vulnerability to execute code in the kernel. SentinelLabs, Max's company, began the disclosure process last year on the 9th of September, and the patch was sent to licensee router vendors on the 4th of October. So it should be incorporated into router firmware updates by now. That's more than 90 days. At this time, SentinelOne has not discovered evidence of in-the-wild abuse.

Okay. So here, in the author's voice, is how this all began. He said: "As a number of my projects start, when I heard that Pwn2Own Mobile 2021 had been announced, I set about looking at one of the targets. Having not looked at the Netgear device when it appeared in the 2019 contest, I decided to give it a lookover. While going through various paths through various binaries, I came across a kernel module called NetUSB. As it turned out, this module was listening on TCP port 20005 on the IP 0.0.0.0. Provided that there were no firewall rules in place to block it, and typical consumer routers don't have any, that would mean it was listening on the WAN as well as the LAN." And he says: "Who wouldn't love a remote kernel bug?

"NetUSB is a product developed by KCodes. It's designed to allow remote devices in a network to interact with USB devices connected to a router. For example, you could interact with a printer as though it is plugged directly into your computer via USB. This requires a driver on your computer that communicates with the router through this kernel module." And of course you don't have to be using this to have it there, alive and running, in your router, if it just has that NetUSB feature which they licensed, the router manufacturer licensed from KCodes.

Okay. He then proceeds to provide a detailed takedown description of his successful hunt for a critical vulnerability in the KCodes code. He discovered a dangerous switch function driven by a command type that's provided by the user, and the rest does not end well. I've provided a link in the show notes for anyone who wants all the gory details. So it's insane. And it is so wrong that this buggy KCodes service is bound to the router's WAN interface. Essentially, 0.0.0.0 is all interfaces on the stack as opposed to if it were bound to 192.168.0.1 or .1.1, that is, bound to the gateway interface. Then it would only be listening on local ports inside the LAN, which is what everyone wants.

Nobody wants this thing listening on the WAN. But it turns out by default it is. That means it is instantly discoverable by bad guys anywhere, and of course by Shodan. It also means that it's instantly testable, as we started out talking about, by any port probe, and I just happen to offer a free online port probing service. So the other link I've provided is a grc.sc shortcut to instantly allow our listeners and anyone to check any router they're behind for this vulnerability.

Open your browser and just put in grc.sc/854, this week's episode number. This will jump you to GRC's ShieldsUP! custom port probe, preloaded to check port 20005. You'll see on your browser screen it sends a bunch of TCP SYN packets spread out over a few seconds. I think it's five seconds. I send one every half second so as not to overload anything and to redundantly send TCP SYNs to make sure that we'll see if we get back a SYN ACK. That will be sent to the IP address also shown on that page to quickly and privately check your browser's publicly exposed WAN interface to determine whether it's accepting incoming TCP connections over port 20005. It should not be. If it is, unplug it. I mean, really, unplug it. Or if you can, add a firewall rule, if your router allows you to, to explicitly block that port on the WAN interface until you're able to update your router's firmware. Hopefully an update is available.

As I said, Max, who discovered and responsibly disclosed this and waited patiently for more than 90 days until last Tuesday the 11th before going public with it, finished his disclosure by writing: "This vulnerability affects millions of devices around the world and in some instances may be completely remotely accessible. Due to the large number of vendors that are affected by the vulnerability, we reported this vulnerability directly to KCodes to be distributed among their licensees instead of targeting just, for example, the TP-Link or the Netgear device in the contest. This ensures that all vendors receive the patch instead of just one during the contest. While we're not going to release any exploits for it, there is a chance that one may become public in the future despite the rather significant complexity involved in developing one. We recommend that all users follow the remediation information above in order to reduce any potential risk."

And we have another example of something good that came from the Pwn2Own competition. Now, let me just say I did read in detail his posting. He did not develop an exploit, but he walked anybody who's competent right up to one. Also, because this is a common code across a large number of routers, all Linux-based, the tools are there for developing it. That means that if you create one exploit, you literally, you're getting millions of devices which are trivial to find. And you're able to execute your own code on those all common Linux platforms. So it's not just one make and model. It's just not one make. It is cross-vendor. This is really going to be juicy.

Leo: Now, to be clear, if it says "closed," are you okay, instead of "stealth"?

Steve: Yes.

Leo: Obviously "open" is bad.

Steve: Yes, open is the bad news. Closed is fine.

Leo: Oh, okay.

Steve: Yeah, closed is fine. All it means is that your router bounced back a no...

Leo: It responded, yeah.

Steve: Exactly. Exactly. Probably sent back a TCP reset saying I got your probe, but I'm not open for business. That's fine. Stealth is cooler. It just means that no response was returned. It's no one's business that there's even anything at that IP listening. Open is the danger.

Leo: And you can't do this from outside your house. You have to do it from within.

Steve: Correct. Correct. I did that. That's clearly the design intent of ShieldsUP! because otherwise it would allow bad people to probe other people's IPs.

Leo: Not good, yeah.

Steve: And that would have been not good, yes. Yeah. So you've got to do it from the LAN, which you want to verify. But just, again, this is not a small thing. Let me please encourage our listeners, grc.sc/854. If you're at work, and you've got a family member or kids at home, give them a call. Have them do this. You definitely want - this thing, there's no way this is not going to be exploited. There's no way because it's Linux, because it's cross-router vendor. There's just no way this is not going to end up getting exploited.

Okay. Chrome is going to be limiting its access to private networks. And this is a win and a half. It will soon be implementing, probably the first of the Chrome, well, the Chrome, Chromium, we can hope that Safari and Firefox follow, a newly proposed web standard which is known as Private Network Access, or PNA. If anyone's interested, I've got a - it's a W3C standard, just like I think it was January 2nd was the date on this, so the ink is not dry, or the HTML not yet set. I've got a link in the show notes.

It will apply new and welcome, in my opinion, controls to block external Internet websites from querying and interacting with devices and servers located inside local private networks. We've talked about this problem a number of times already. As I mentioned, this change will occur as Chrome implements this new W3C spec known as Private Network Access. It'll be rolled out in the first half of the year for Chrome. PNA adds a mechanism through which external Internet sites must first ask systems inside local networks for explicit permission before being allowed to have any sort of connection. Okay, now, that's what some of the coverage says, and it's not exactly the way this works. What happens is Chrome will notice when an external Internet-based server is doing something that requires internal access. And then Chrome will take responsibility for saying, wait a minute. Let's see if that's okay.

Okay. So Chrome and any other PNA-compliant browsers will first send what's known as a "CORS preflight request." CORS is Cross-Origin Resource Sharing, a nice addition to the web standard for controlling who gets to talk to whom cross-origin. So Chrome will first send what's known as a "CORS preflight request" to the local server or service, whatever it is on the LAN, before granting any Internet-originating request for a private network resource. This CORS request will ask for and needs to obtain explicit permission from the internal target server or service. The preflight request, which is just a fancy name for an extra header, is a new header. It is Access-Control-Request-Private-Network: true is what Chrome sends out.

And the response to it must also contain a header: Access-Control-Allow-Private-Network: true. If the targeted local devices such as servers or routers fail to respond, Internet websites will be blocked from connecting, as they never should probably be allowed to anyway. I can't imagine what that's used for. Maybe there is a legitimate use. We know it's been exploited a lot. So this is a wonderful improvement in cross-origin access control.

As we know, and have devoted a number of podcasts to explaining, bad guys have figured out that they can use a browser as a proxy to relay connections to an individual's or a company's internal network. For example, a malicious website could contain code that tries to access an IP address like 192.168.0.1, which will often display the LAN router's local admin logon page, which is only accessible by design from the router's LAN interface. But because the request is coming from the user's browser on the LAN, the router assumes that the user wants to log in. So as we've seen, when unwitting users access a malicious site, it's been possible to induce their browser to make a request to their router without their knowledge. This can send malicious code to bypass the router's authentication and modify router settings. It's happened.

Variations of these Internet-to-local network attacks could also target other local systems such as internal servers, domain controllers, firewalls, or even locally hosted applications. So by introducing the PNA specification inside Chrome, and within Chrome's permission negotiation system, Google will be moving to prevent such automated attacks. And I say yay.

According to Google, a version of PNA has already been slipped into and shipped with Chrome 96, which was released back last November. But full support will be rolled out in two phases this year, with Chrome 98 in early March and Chrome 101 scheduled for late May. In this coming March's Chrome 98, Chrome will begin sending these preflight requests ahead of private network subresource requests. These preflight failures will only display warnings in the DevTools of the browser, without otherwise affecting the private network requests, so they remain allowed and unblocked. So this, you know, this gives devs who are wanting to bring up support for this affirmative information about the fact that this is going on.

Chrome gathers compatibility data and reaches out to the largest affected websites. So Chrome will be using this for instrumenting their own research. And if it turns out that there are major websites doing this, they'll notify them that, hey, you know, this is going to start getting blocked here before long. Google expects this to be broadly compatible with existing websites. And, yeah, why wouldn't it be? It doesn't break anything yet.

Then, no earlier than late May's scheduled Chrome 101, assuming that established compatibility data indicates that the change will be safe enough and that sufficient outreach has successfully occurred, Chrome will begin enforcing that preflight requests must succeed, and it will otherwise fail those externally sourced requests. They said a deprecation trial will start at the same time to allow for websites affected by this phase to request a time extension. The trial will last for at least six months.

So the concern that's evidenced here is that, as is always the case, tightening up security may break something that was happening, little known and unseen, in the background. But overall, this is a welcome improvement. If there is something on the LAN that really does intentionally wish to be able to receive and respond to requests originating from the user's browser, but triggered from an external source, then such devices will need to be updated with an awareness of these new forthcoming PNA controls. And that simply entails replying with the newly added header to continue enabling what had always been allowed before. So not a big change to make. Easy to do. And I just think this is a big useful step forward. And actually it was a couple of Google guys who apparently were responsible for getting this thing through the W3C. So props to them.

I'll just finish with three high-severity flaws, you know, just mentioning them, in WordPress. As I mentioned at the top of the show, as a consequence of these three, 84,000 WordPress-based websites are affected, and they're very serious. The guys at Wordfence titled their disclosure "84,000 WordPress Sites Affected by Three Plugins With the Same Vulnerability." They wrote: "On November 5th, 2021" - so November of last year - "the Wordfence Threat Intelligence team initiated the responsible disclosure process for a vulnerability we discovered in 'Login/Signup Popup,' a WordPress plugin that is installed on over 20,000 sites. A few days later we discovered the same vulnerability present in two additional plugins developed by the same author: 'Side Cart WooCommerce,' installed on over 60,000 sites; and 'Waitlist WooCommerce,' the back in stock notifier which is installed on over 4,000 sites. This flaw made it possible for an attacker to update arbitrary site options on a vulnerable site, provided that they could trick a site's admin into performing an action, such as clicking on a link."

They said: "We sent full disclosure details on November 5th, 2021, after the developer confirmed the appropriate channel to handle their communications. After several follow-ups, a patched version of 'Login/Signup Popup' was released on November 24th, while patched versions of 'Side Cart WooCommerce' and 'Waitlist WooCommerce' were released on December 17th." So they waited 90 days again, and now they're, well, December, yeah, not in that case. But they gave time to get these things pushed out and updated.

They said: "We strongly recommend ensuring that your site has been updated to the latest patched version of any of these plugins, which is version 2.3 for 'Login/Signup Popup,' version 2.5.2 for 'Waitlist WooCommerce,' and version 2.1 for the 'Side Cart WooCommerce' at the time of the publication." So anyway, just a heads-up. When I hear that something like WooCommerce is running on WordPress, wow, you know, we hear about commerce sites being attacked all the time, and WordPress is not a place you want to do something that is really security sensitive. Blogs and comment scrolls and comment threads, that's one thing. But more than that, I don't know.

Okay. A couple closing-the-loop pieces. Peter Morelli said: "'Expanse' season complete. You'll love it." So I just wanted to note to our listeners who are interested, if you've been waiting before you jump on the final season of "Expanse," I have been waiting. So Peter, thank you for that. I wanted to let everybody know.

Chris Miles, or Milesey, he said: "Steve, have you guys seen the huge QNAP vulnerability that has ended up with hundreds of thousands of units infected with ransomware code? Even a unit of mine with strong passwords was hit. Thankfully, one-way versioned backups meant nothing was lost." And I'll just - I wanted to take this opportunity. Thank you, Chris, for the heads-up. I keep trying to get to it in the show, but there's been so much else to talk about.

You know, we said before there is available non-QNAP open source software that looks like it does a much better job. QNAP just perennially seems unable to get their act together here. They've have one serious remote code execution problem after another.

So at some point just go find the - I don't remember if it's FreeBSD based, or OpenBSD, or what it is. But it's one of them that you can run on the QNAP system in order to do network attached storage features and just get away from QNAP's solution. Hardware, yes. Software, apparently not.

Okay. Two listeners via DM, so I didn't put their name in since DM is inherently private, and I wasn't sure they wanted me to disclose who they were. First one said: "Hi, Steve. I was intrigued by your mention of refilling the SodaStream bottles in last week's SN-853. I use SodaStream a lot to replace buying soda," he says, "(called 'Sparkling Water' here in the oh-so-pretentious U.K.) in plastic bottles, but the refills" - meaning of SodaStream - "are expensive. Do you refill from brewing gas bottles or some other source?"

And Listener 2: "Hi, Steve. Would you be so kind as to share the link to the SodaStream refill adapter you're using? I would very much appreciate it. And of course I will keep this confidential." I apparently will not, although I did keep his name confidential. So anyway, for those who don't know, this interest was stirred up by Leo and me talking about the fact that I have been for years refilling my empty SodaStream steel CO2 canisters. For a while I was taking them to my local, what is it, Ace Hardware was a provider. You know, they talk about how easy it is to swap the canisters.

Well, you take your empties back, and it may be easy, but it sure ain't inexpensive. And after a while, I don't know how it occurred to me, but I thought, you know, I wonder if there's a way to refill these. So a little bit of googling, sure enough. There is a common source of food-grade CO2. You want to make sure you get, like, food-grade everything. And that's people who do home brewing. It turns out about a mile from me is a nice little home brewing retailer on Bristol Avenue in Costa Mesa. And I guess it's more than a mile, but still it's convenient. And home brewing people use CO2. And so on Amazon I purchased first a 20-pound CO2 canister with a siphon. That's important. The siphon is a key, unless you want to have your big canister upside down.

The siphon just - it's just a little tube that runs down to the bottom of the tank so that it's pulling liquid CO2 from the bottom out the spigot as opposed to taking the gas off the top. So you get that. That's $150 if you're interested. And then a refill adapter is nothing but a cleverly simple threaded, it's female threads on both sides. So you screw one end. It's got the proper threads for the standard CO2 connection. And then on the side that's facing outward is the threading that matches the universal SodaStream bottles. And so basically all this does is it just mates a now-empty SodaStream bottle to the liquid CO2-filled big 20-pound refillable at your local source of CO2 bottle.

Leo: I would just caution people, because I did this, bought all this, and I'm still looking. But Russell, who is a home brewer, says it used to be he could bring his bottles in, and they would fill it. Now they exchange them at the places he's gone, for some crap tinted up bottle, or they require this whole testing process before they'll do it. They say leave it here, and you can come back tomorrow, that kind of thing. So I would check, before you jump whole hog into this, to make sure you do have a good source of CO2 that you can use to refill it. You know, call around.

Steve: Very good. Very good, Leo.

Leo: Because apparently, you know, and I'm not surprised, these guys, they're kind of - maybe because of COVID, they're tired of dealing with people. I don't know. It's getting maybe a little harder to get these filled.

Steve: That's interesting.

Leo: You're lucky, you've got somebody who can do it.

Steve: Yeah, and they're nice people, and they're glad to see me.

Leo: Bring them candy. Yeah, exactly, they like you now, so they trust you, yeah.

Steve: So I thought I would share a couple pointers, and then we won't talk about this again. First is you absolutely want your SodaStream empty canister frozen. That is, what we do is we have like a spare freezer, and so it's got our empties in there. You'll want to refill them when they're cold. That minimizes the re-expansion of the liquid CO2, and you get a much fuller bottle. The second is that the top of the SodaStream, basically it's got a little push valve that when you push the SodaStream down, a pin pushes down in the center in order to release the compressed CO2.

Leo: They charge so much money for what is essentially a mechanical lever, a plastic thing with a mechanical lever, because all it's doing is pushing in the button, and it goes in the spout, and it goes into your soda bottle.

Steve: That's right. However, the one gotcha here is that if that is pushed too far in, it stops again. So the trick of refilling these is to bleed it in. If you hook everything up and just open the main valve, nothing happens. You'll hear, like, eh, and then it's like stopped. So what I've learned is, first of all, cool the receiving canister. And then just ever so slightly twist the valve open, and you'll hear the sound of gas or fluid flowing. That's all you need to do. Walk away, as I mentioned jokingly but actually sincerely last week. Don't stand directly in front of it. That's just, I mean...

Leo: Especially at crotch height.

Steve: It is at crotch height, so...

Leo: Could impede your propagating, so don't do that, yeah.

Steve: It would limit even practicing.

Leo: Yeah.

Steve: Yes. So, and it takes, you know, a couple minutes. And if it slows down, then you can creep the valve open again and restart the little "eeeeee" sound. But, boy, it's just so gratifying. When you're unable to put any more in, you turn off the main valve, remove the cylinder. It is heavy because it is now filled with liquid CO2, and then you're good to go for a whole long - and I think I paid $14 to have this 20-pound big, big floor-standing canister refilled, which beats the hell out of going and buying new canisters from, I mean, and because we love SodaStream.

Leo: Oh, yeah. The idea is to save money, in the long run anyway. So, yeah. And eliminate single-use plastic bottles.

Steve: Yes.

Leo: That's why we got it, yeah.

Steve: Okay. So Sunday evening, night before last, after a weekend spent scratching my head and experimenting with an older motherboard which I had purchased through eBay in order to duplicate what at least one of our testers had seen, I figured out what was going on and adjusted SpinRite's core technology to accommodate it and any other similar systems. The trouble I had been having was occurring due to Intel's 82371 PCI-TO-ISA / IDE XCELERATOR, as they call it. It's known as the PIIX4. And the chip's spec is dated April of 1997. So, yeah.

Getting this v6.1 release of SpinRite to run everywhere is turning out to be far more work, and I'm not surprised, since the whole point of 6.1 is to bypass the BIOS and talk directly to the hardware. The BIOS may not have been terrific for performance, it did not have performance on its side, but it does have compatibility. I knew that this was likely to be where a lot of time was going to be spent. Fortunately, we have a really amazing group of development testers. The instance of GitLab that I brought up in mid-December currently has 241 registered participants. So by far, most of them watch and silently test the code as it evolves. If it does something wrong, then I'll hear about it. But it's very gratifying to know that SpinRite is receiving this level of pounding at this stage.

Yesterday, or day before yesterday, rather, in that end-of-the-weekend's work update, I posted to the newsgroups. I said: I think that with the Supermicro, the Asus EEE PC 901, and now millQ's 82371 chip issues all resolved, that's the last of the big mystery behavior problems. This just leaves me with a bunch of less interesting and already understood things to fix and clean up. Once those are finished, I think we'll be ready to thoroughly pound on what we have to see whether anything else falls off. It feels like we're getting close to having this operational foundation fully functional. So that will be a major milestone.

We will get to a point where the code I have is identifying every drive that everybody has connected using every controller on every system they have correctly and running benchmarks on those to demonstrate that it's able to talk to the drive. And part of that is a read/write test which I perform way out at the far end of the drive. And at that point it's just time to basically finish SpinRite. That is, all of this hardware compatibility layer will be resolved, and I get to move forward, you know, update the screens to hold larger numbers and that kind of thing. And so it'll be a nice milestone to get to. And we're probably within a week or two of that. So yay.

Leo: I see the SodaStream bottle.

Steve: That was my SodaStream bottle I was drinking from.

Leo: I see Steve's SodaStream. And of course he has the world's largest SodaStream bottle.

Steve: Okay. So many security firms are tracking threat actors who immediately and predictably jumped aboard the Log4j bandwagon. You know, it's been a feeding frenzy for the security firms. To help bring this home and make it a bit more real, I wanted to share a piece of Check Point Research's reverse engineering work on a typical threat the Internet is now facing. I've got, for anyone who wants more detail, as always, a link in the show notes.

Last week, Check Point documented the efforts of an Iranian government-backed group known, again, not just Iranians, Iranian government-backed group known as APT35, also known as Charming Kitten, TA453, and Phosphorus. This group started widespread scanning and attempts to leverage the Log4j flaw in publicly facing systems only four days after the vulnerability was disclosed. And all the bad guys knew, now that this was public, it was going to get remediated at some speed, the point being let's be first. You know, get in there before the back doors get closed.

Since this particular actor's setup was hurried, they simply grabbed one of the publicly available open source GitHub-hosted JNDI exploit kits. Yes, they were on GitHub initially. But that kit has been removed from GitHub due to its enormous popularity following the vulnerability emergence. Why bother reinventing that particular wheel when time is of the essence? They also based their operations upon their preexisting infrastructure, rather than like creating a whole new one, and that infrastructure was already well known to Check Point, thus making its detection and attribution all the easier.

In the show notes I have a flow chart which shows the path that the exploit takes. And it could hardly be any easier or direct. First, the attackers send a crafted request to the victim's publicly facing Internet-exposed resource, whatever it is, a server of some sort. In this particular case the weaponized payload was sent in through either the user-agent or the HTTP authorization headers. Remember that all that needs to happen is that something somewhere that's Java-based logs part of the query that contains this weaponized string. In order to log the query, Log4j examines what it's logging, sees a JNDI component, and goes about its job of obtaining the content from the LDAP URL contained in the query which is being logged.

So the vulnerable machine, as it's been instructed to do basically, although not after it's been patched, but until then, reaches out to what they labeled in their diagram a Log4j Exploitation Server, which assembles and returns a malicious Java class which will be executed on the vulnerable machine. The class runs a PowerShell command with a Base64-encoded payload. And I actually have a picture of the actual payload, the exploit dot command, powershell, and then the encoded payload. That PowerShell command downloads a PowerShell module from an Amazon S3 bucket URL, and it actually is https://s3.amazonaws.com/doclibrarysales/test.txt, and executes it. And we have a picture of that in the show notes, the actual thing that's downloaded.

The downloaded PowerShell payload is the main module that's then responsible for basic communication with the command-and-control server and the execution of additional modules which may be received. So the main module performs the following operations. It validates the network connection. Upon execution, the script waits for an active Internet connection by repetitively making HTTP POST requests to google.com with the parameter hi=hi, just to see if it can succeed. That's how it detects whether or not it's got an Internet connection. Assuming that it does, it knows that.

It also performs basic system enumeration. It collects the Windows OS version, the computer's name, and the contents of a file Ni.txt in $APPDATA, in the $APPDATA path. The file is presumably created and filled by different modules that will be downloaded by the main module. It then retrieves the command-and-control server's domain. The malware decodes the command-and-control domain retrieved from a hardcoded URL located in the same S3 bucket from where the backdoor was downloaded. So the bad

guys have dynamic control over that by deciding what goes in this AWS bucket. It also retrieves, decrypts, and executes follow-up modules.

Okay. So once all the data is gathered, the malware starts communication with the command-and-control server at the domain which it determined by pulling that from the Amazon AWS cloud bucket. And it does that, it communicates with the command-and-control server by periodically sending HTTP POST requests, I mean, none of this is high tech. None of this is rocket science. This is easy to do, which is why this terrified everybody so much. So this thing sends HTTP POST requests to a pre-configured URL with each POST request containing information from which to build a session key: the OS version, the computer's name, and the contents of that file in the $APPDATA directory. So that ends up being something unique which it uses to identify itself each time. And I think as I recall it puts it in a session header in the POST query.

In response to the command-and-control server's receiving these POST requests, it can either choose not to respond, in which case the script will keep sending POST requests periodically to continue to provide the server with a stream of response opportunities, or the server will return a Base64-encoded string. Now, just as a reminder, Base64 is a means for sending binary data over an ASCII channel, that is, over a text-only channel. Groups of three 8-bit binary bytes, so three 8-bit binary bytes is 24 bits, they're regrouped from three 8-bit bytes to four 6-bit bytes. Six bits can have 64 combinations.

So we take the lower and the upper alphabet, gives us 2x26 characters, or 52 characters. We add the 10 decimal digits. That brings us up to 62 characters. And then we toss in two additional ones, the plus and the forward slash, which brings us to 64. So in groups of three, binary is taken from the source binary. Those 24 bits are regrouped into four characters, each one of 64 different possibilities. That's then all munged back together and sent down with the client, which reverses the encoding process to restore the original binary. This allows the malicious server to squirt anything it wants into the victim machine that's making the queries. The modules downloaded in this fashion are either PowerShell or C# scripts.

The modules sent by the command-and-control server are executed by the main module, with each one reporting data back to the server separately. So the original module comes in, looks around, sets up shop, figures out who to talk to, initiates the dialogue, and does that periodically. If in response to one of its multiple POST queries it receives a blob of Base64, it goes, oh, okay, something to do. It decodes it back into whatever it was before, you know, removes the Base64 encoding - we know that that's going to be a PowerShell or a C# script - and runs it.

At that point that subsidiary module takes off on its own, and it establishes its own communication directly with the command-and-control server. The command-and-control cycle continues indefinitely, which allows the threat actors to gather data on the infected machine, run arbitrary commands, and possibly escalate their actions by performing a lateral movement or executing follow-up malware such as ransomware. In other words, this thing can do anything it wants to, once it gains a foothold.

So the modules. Every module is auto-generated by the attackers based on the data sent by the main module. Each of the modules contains a hardcoded machine name and a hardcoded C&C domain. Every module Check Point observed contained a block of shared code, which makes sense because there's a bunch of stuff that they're all going to do regardless of their specific function. And that is encrypting the data to be sent; exfiltrating the gathered data through a POST request or uploading it to an FTP server, that also happens; and sending execution logs to a remote server.

In addition to this, each module performs one specific job, that is, in addition to those things they all have in common. Check Point retrieved and analyzed modules for six

different functions: listing installed applications, that is, applications installed on the machine; taking screenshots; listing the running processes; getting OS and computer information; executing a predefined command from the command-and-control server; and then, finally, cleaning up any traces created by any of the other modules.

The applications module uses two methods to fetch and return a list of installed modules. It can either enumerate the Uninstall registry values or use the Windows Management Instrumentation command in order to get an enumeration. It gets those, encrypts them, and sends them back to headquarters.

The screenshot module, they found both C# and PowerShell scripts for the screenshot. They both have the capability to capture multiple screenshots at specified intervals and upload the resulting screenshots to an FTP server whose credentials are provided by the script. The C# script uses a Base64-encoded PowerShell command to take the screenshot from multiple screens. So again, you might have this thing in your computer, not know it. You're doing things, and this thing is spying on you, sending shots of your screens back to headquarters. The processes module obtains a list of the machine's running processes using the tasklist command, gathers them, encodes them, sends them back.

The system information module contains a bunch of PowerShell commands. What was interesting was that in the instances that Check Point saw, the bad guys had commented out all of these potential sources of information. They just weren't using it. This told Check Point that this whole campaign was hastily assembled since the entire, as we know, attacker community was well aware that systems would be closing their doors very quickly. So there were all these different suggestions of the moment this thing went public the attackers jumped on it and said let's quickly get something together that we can exploit this with.

And finally, we have the command execution module, which is able to essentially download and execute any actions, any commands that are provided by the command-and-control server. They saw, for example, listing the contents of the C: drive root; listing the specific WiFi profile details using netsh, the WLAN subcommand of that. And also listing all the drives using Get-PSDrive, a PowerShell enumerator.

And finally, the cleanup module. It's dropped after the attackers have finished their activity and want to remove any traces that they've been inside the system. The module contains cleanup methods for persistence-related artifacts in the registry and the startup folder, any files created, and any running processes. It contains five hardcoded levels of sort of like stages of cleanup, depending upon the stage of the attack, each one serving a different purpose. Check Point said that the design and the intent of the cleanup module made it clear that the threat actors want to keep the infection on the machine, first of all, for as long as they deem necessary. But then once their goal has been achieved, they want to disappear without a trace so that no one believes that an attack occurred.

As for attribution, of course we know attribution of network remote attacks often falls somewhere between difficult to impossible, but not so in this case. Most advanced persistent threat actors put some effort into making sure to change their tools and their infrastructure to avoid being detected in the first place and to make attribution much more difficult if they were detected. And in fact we know that the SolarWinds attacks were famous for, like, really working to obscure the path by which the infection happened if it were to be discovered. However, APT35 does not conform to this behavior. Apparently the group is famous within the cybersecurity community for the number of operational security mistakes they've made in previous operations, and they tend not to put too much effort into changing their infrastructure once it's been exposed. So it's little wonder that their operation, as Check Point has detailed it, has significant overlaps in the code and the infrastructure which previously identified the activities of APT35.

As for code overlaps, four months ago, in October of 2021, Google's TAG team, remember, their Threat Analysis Group, published an article about APT35's mobile malware. You know, because Google and Android. Even though the samples Check Point analyzed were PowerShell scripts - meaning PowerShell as opposed to Android, so Windows only - the similarity of coding style between them and the Android spyware that Google attributed to APT35 immediately caught Check Point's attention.

For one thing, the implementation of the logging functions was identical between the Android App which Google analyzed and this present campaign's PowerShell modules which use the identical logging format, even though the commands are commented out and replaced with another format. The fact that these lines were not removed outright, Check Point felt, might indicate that the change was done only recently. And the syntax of the logging messages themselves being logged is identical.

As for infrastructure, both then and now campaigns, October and now, apparently use the same server-side infrastructure. When a client POSTs data to a remote HTTP server, the server-side path of the query is called the "API endpoint." Google's mobile analysis and Check Point's both revealed the use of the common endpoint "/Api/Session." Now, okay. That's not a high-entropy name. Could have just been a collision of convenience. But Check Point felt encouraged by the observed overlap, and they stated in their report that other API endpoints are similar but not entirely identical due to the differences in the functionality of the platform. So didn't make sense for them to be completely identical.

Check Point also observed that not only are the URLs familiar, but the command-and-control domain of the PowerShell variant responds to the API requests that are used in the mobile variant. This suggests similar, if not identical, server-side support for both campaigns.

So Check Point concluded its report by observing that every time there is a new published critical vulnerability, the entire Infosec community holds its collective breath until its worst fears come true. Scenarios of real-world exploitation appear, especially by state-sponsored actors. As they demonstrated in their report, the breath-holding wait in the case of the Log4j vulnerability was only a few days. The combination of its simplicity, its publicly available open source code samples, and the massively tantalizing number of vulnerable devices made this a very attractive vulnerability for actors such as APT35. And I have no doubt that, while I don't think I will continue giving this in-depth coverage because we know pretty much everything there is to know about it, well, if something major happens, it'll certainly be newsworthy. But that's how this stuff works. Again, just it's frightening how non-rocket science, how script-kiddie level this thing is, and that it can get up to so much mischief.

Leo: It's amazing. Even a, what is it, a kitty, what kind of kitty, Charming Kitten? Even Charming Kitten can do it. Who comes up - is that like a Vulnonym? Who comes up with, I mean, there's Fancy Bear for the Russian group, Charming Kitten for the Iranian group. Somebody's coming up with these.

Steve: Yeah, I don't know.

Leo: Must be the CIA or the NSA. That's just wild. Steve, you did it again. Here we are at the end of another fabulous episode. If you would like to know more, Steve has the show notes at his website. It's not all that's there. There's a lot of stuff at GRC.com, including that ShieldsUP! test where you could test your port 22009? What was it? Some...

Steve: Please do. 20005.

Leo: 20005.

Steve: So it's grc.sc/ our episode number, 854.

Leo: Okay. Actually, every time I set up a new router or a new network, I use ShieldsUP! to check, make sure it's properly configured. It's a very useful tool. One of many things Steve gives away. The only thing he doesn't, that's his bread and butter, SpinRite, his very, very popular mass storage maintenance and recovery utility. If you don't have a copy, you really ought to have one. If you get 6.0, the current version right now, you get a free upgrade to 6.1. It's imminent. You can help in the development of it, as well. There's a forum there. All of that at GRC.com.

16Kb versions of the show audio, as well as 64Kb audio versions of the show are there. The transcripts, which are very handy if you like to read while you listen, or just to search through to find a particular part of any given show. All at GRC.com, and lots of other stuff, as well.

We have 64Kb audio versions and full video, as well, if you want to watch, available at our site, TWiT.tv/sn. There's a YouTube channel dedicated to Security Now!. All the videos, all the time. You can also subscribe in your favorite podcast player, because it is a podcast, and download it automatically. Or even go to the website, TWiT.tv/sn, and download it from there.

If you subscribe in a podcast player, you get it automatically, which is probably a good idea because this is one of those shows where I think you want all the episodes; right? You can go back at the website to Episode 1 and come forward from there. The feeds only have the last 10 episodes for reasons of economy, but that'll get you started, anyway. So look for Security Now! in your favorite podcast player.

If you want to watch us do the show live, we stream live at live.twit.tv, 24/7. This show is every Tuesday, 1:30 Pacific, 4:30 Eastern, 21:30 UTC. You can watch live, listen live, chat live at irc.twit.tv or in the Club TWiT Discord server, which is a lot of fun to be in there. And I think that pretty much covers it all. We'll be back next Tuesday with another thrilling, gripping edition of Where the Hackers Are.

Steve: Right-o.

Leo: Thanks, Steve. We'll see you next time.

Steve: Thanks, buddy.