A Critical Windows RPC RCE

Description: This week we examine Chrome's third zero-day of the year, followed by Microsoft's massive 128-patch fest last week, and we note that we don't even bother counting Windows zero-days, though there were another two this month amid the 47 critical vulnerabilities that were patched, one of them being so worrisome that it captured this week's podcast title, which we'll cover at length before we conclude. We also have more WordPress add-on trouble, the return of a longstanding problem in Apache Struts, and we have some interesting commentary about the current hackability status of the United States nuclear arsenal. I want to share a bit of closing-the-loop feedback with our listeners and give everyone a snapshot into the recent work on SpinRite. Then we're going to take a close look at the one flaw, out of 128 that Microsoft patched last week, that truly has the entire security industry on pins and needles because it enables a zero-click Internet worm.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-867.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-867-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. There is another Chrome zero-day, its third of the year. But that's nothing compared to Microsoft's 128-patch fest last Tuesday. Steve takes a look at some of them, including 47 critical vulnerabilities, and one he'll take a look at in greater detail, the RPC RCE. Plus, yes, another problem with WordPress plugins. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 867, recorded Tuesday, April 19th, 2022: A Critical Windows RPC RCE.

It's time for Security Now!, the show where we protect you, your loved ones, your privacy online with this guy right here, Steve Gibson. It's a Tuesday. It must be Stevie G. Hi, Steve.

Steve Gibson: Yes, indeed. In fact, this is the third Tuesday of the month, which makes it the Patch Tuesday retrospective edition. Oh, and, boy, we have really - in fact, there was one that was so bad that it became the title of the podcast. We're going to examine Chrome's third zero-day of the year, followed by, as I said, as you led me into, Microsoft's massive 128-patch fest last week. And we note that we don't even bother counting the Windows zero-days. Yes, it's a big deal that it's Chrome's third. Windows, eh, why bother counting? They had two this month alone. And that was amid the 47 critical vulnerabilities that were patched, one of them being so worrisome that it captured this week's podcast title, which we'll cover at the end before we conclude. But we also have more WordPress add-on troubles, the return of a longstanding problem in - and

everybody knows this term - Apache Struts, which of course gave Equifax some serious heartburn back in 2017.

We got some interesting commentary about the current hackability status of the United States Nuclear Arsenal, which of course is sort of an issue now that Putin's been rattling his sabers. I want to share a bit of closing-the-loop feedback with our listeners and give everyone a bit of a snapshot into the recent work on SpinRite, which I haven't mentioned for a few weeks. Then we're going to wrap by taking a close look at the one flaw out of the 128 that Microsoft just patched last week that truly has the entire security industry now on pins and needles because it enables a zero-click Internet worm.

Leo: Whoa, that's not good.

Steve: That'll ruin your day, or some people's days.

Leo: Something, yeah.

Steve: And we do have a fun Picture of the Week, as always.

Leo: Okay. Picture of the Week time, Steve?

Steve: I wonder if .cloud is a top-level domain? That would kind of make sense.

Leo: I feel like there is. That's a good question.

Steve: Yeah, then they could be cloud.jumpcloud.cloud.

Leo: That's a few too many clouds. All right. I'm ready for the Picture of the Week.

Steve: So I was saying to you before, looking at this Picture of the Week, that this made me ask myself if I like physical humor. Because, you know, one of the things I got the biggest kick out of, one of our like legendary Pictures of the Week, was that ground wire stuck into a pail of dirt.

Leo: You love this stuff. I know.

Steve: You know?

Leo: You love this stuff.

Steve: It's like it's grounded. And then the other favorite one of mine was when the hardware store had bolt cutters which it was securing against theft by looping a completely bolt cutter-able security cable through the bolt cutters. It's like, uh, you

know, the bolt cutters are designed to cut the cable that you're using to hold them. So I don't think that works.

Anyway, this one is sort of another. We have like the third one here. This is like, it looks like an old-school industrial control panel with big indicator lights, and it's got those labels that were engraved where the top plastic layer of the label is white, but it's on a black plastic backing so when you engrave it, the black shows through so they can't fade. They're like made forever.

Leo: It looks like it's Soviet era technology, is what you're saying.

Steve: Yeah, okay, perfect, yes. And the controls are these big clunk switches that you would turn left or right, rotate left or right in order to engage the dump of the bad fluids or whatever out of some, you know, turn the pumps on and then the lights dim a little bit. Everyone's like, oh, the pump's on.

Leo: Good, satisfying chunk when you turn it, yes.

Steve: Anyway, somebody wanted to make absolutely sure that one of these twisty knobs would never be twisted, so they took a big, is that a crescent wrench or a plumber's wrench? I'm not sure exactly.

Leo: Yeah, a pipe wrench, yeah.

Steve: A pipe wrench. And it looks like it's been around for while because it's kind of rusty.

Leo: It's pretty rusty. I know, I love that.

Steve: Yeah. And so it kind of completes the picture. And so they first screwed this wrench down onto the handle of this twisty knob so that it was not going to move anymore.

Leo: It cracks me up.

Steve: Then they took a piece of metal, kind of sheet metal strapping tape, I guess, but metal strapping tape, and bent it over the handle of this wrench and then screwed it to the front of the panel. And, you know, as I'm looking at this, I'm thinking, could you not have opened up the panel and jumped across that switch if you wanted it to always be on, or disconnected it if you wanted it to always be off. But apparently that was beyond these people, who decided instead we're going to screw a permanent wrench onto the outside of this and then affix it to the panel with some sheet metal strapping tape. It's like, okay, well, it did make for a great Picture of the Week, so...

Leo: It's really bizarre.

Steve: And somebody just tweeted it to me in the last couple days.

Leo: It's very funny.

Steve: So thank you very much for that.

Leo: Love it. Love it.

Steve: To the rest of our listening audience, you now know where my funny bone is. It is in a pail of dirt that has a ground wire stuck into it, which that's classic, and similar ilk. I also love those water flow ones where like the pipe is broken, but somehow the water is going through the air into a hole that someone cut in the pipe in order to get the mission to be accomplished. Anyway.

Okay. So on our zero-day watch, last Thursday Chrome received an emergency update bringing desktop Chrome to v100.0.4896.127. And that's for Windows, Mac, and Linux desktop Chromes. The exploit is once again a type confusion flaw which we've talked about in detail before so I won't go into it again, also found in Chrome's V8 script processing engine. And this one was found by Google's own Threat Analysis Group. But it's a zero-day because they found it in use. Somebody - and I love it when their description says this could do this or could do that. Well, you think? Because someone was actually using it to do exactly that. So yes, it's not that, if exploited, it could do this. It's like, well, we watched it doing that and were embarrassed. We're sorry that that's the case, but update Chrome.

Anyway, this is the third this year. It's the second type confusion flaw. And the first one was a use-after-free in Chrome's animation support. And as usual, users of Chromium-based browsers - the other ones, right, Edge, Brave, Opera, and Vivaldi - will be getting updates, probably have hopefully already, since this thing would have been able to be leveraged against any Chromium-based browser. And Leo, I've heard you several times lament this monoculture, the browser monoculture that we're slipping into. And here's another like perfect example is the bad guys see that everybody now except Safari and Firefox are going to be victims to a single flaw, if they can find it. Which of course is the liability. The good news is, when you fix one, you fix it for all the others. But on the other hand, they were all vulnerable in the first place. So still, a browser monoculture is a bad place to be.

Anyway, Google never tells us more about this because why would they? They say that they're not going to talk about it until everybody's been well patched. And then by then no one cares. So they don't talk about it. So basically never. Okay. This is the official name of the week after Patch Tuesday has become Follow-Up Tuesday because in this case we get to talk about what happened. And this one won't disappoint. Users of Windows received a total of 128 - this is like, I think it's since 2020 was the previous month that had this number of problems fixed all at once, 128 fixes for known security vulnerabilities pretty much everywhere - desktop and server, Defender, Office, Exchange Server, Visual Studio, print spooler. Windows DNS Server had a bunch we'll talk about. And, you know, "V" everything.

So yeah, pretty much if you're going to have 128 fixes, at least they're well distributed. Of the 128 bugs which were addressed, 10 were rated critical, with a few even earning that rare 9.8 CVSS, which frankly is fortunately rare for Windows. We don't actually see a lot of 9.8s in Windows. This time we got a couple. We'll get back to those in a moment.

A whopping 115 of the others - so we had 10 that were critical. 115 others were important. And the final three were considered to be moderate. As I mentioned, we got a pair of zero-days, one that's known to be actively exploited in the wild, so that we kind of call a real zero-day, and then what Microsoft calls a zero-day because even though it hasn't been seen in use, somebody talked about it, and that upset them. It's been publicly disclosed.

So the spread of these problems break down as follows. There are a total of 47 remote code execution vulnerabilities. Okay. Just take that in for a second. 47 remote code execution vulnerabilities that they have fixed, and the same number of elevation of privilege vulnerabilities, also 47. And remember, lest anyone thought that troubles with Print Spooler were all behind us, 15 of those 47 privilege escalation or elevation vulnerabilities were found in Print Spooler. There were also 13 information disclosure problems fixed, nine denial of service, and three spoofing vulnerabilities. And aside from all that, Edge also separately had 26 flaws fixed. You know, and Chrome hasn't had that many. So as we know, Microsoft is kind of trying to Edge-ify the Chromium core. Sounds like or looks like maybe they're having some problems doing that securely.

The flaw that's being actively exploited, that is, the true zero-day, is an elevation of privilege vulnerability occurring in the Windows Common Log File System. And that one, even though it's the zero-day, is not one of the 9.8s. It's a 7.8. And it was, interestingly, found and reported both by the NSA, the U.S.'s National Security Agency, and some researchers at CrowdStrike. So several people saw that in use and said, uh, Microsoft, there's a little problem over here in the Windows Common Log File System.

Now, the other zero-day which has been publicly disclosed, but is not known to be under active exploitation, most likely because it's difficult to leverage, is also a privilege escalation. This one occurs in the Windows User Profile Service where actually we've been seeing some problems recently. Since its successful exploitation requires an attacker, they said, to win a race condition where timing is critical, that probably explains its somewhat lower CVSS of 7.0, and probably also why, although it's been talked about publicly, it hasn't, as far as we know, been used, probably because it's tricky to somehow win that race condition.

However, there are two biggies among the crop, both coming in with, as I said, rare for Windows CVSSes of 9.8. The least worrisome of the two is in Windows Network File System. And the other 9.8er that has the security industry, as I said at the top of the show, a bit on edge is a remote code execution flaw in the often exposed Windows Remote Procedure Call Runtime Library. That's the RPC of this podcast's title, Remote Procedure Call. Other RCE flaws, remote code execution flaws, were fixed in Windows Server, Windows SMB, and Microsoft Dynamics 365.

DNS, Microsoft's DNS server. We touched on this in a couple podcasts ago, so that suggests that people had been looking at it more recently. And as we have seen, it's often the case that when attention gets turned to a specific item in Microsoft's repertoire, lots of problems get found there. Well, that's the case here again. The well-known security researcher Yuki Chen focused on DNS. Microsoft credited Yuki with discovering and responsibly disclosing a total of 18 previously unknown problems in their DNS server, one being an information disclosure flaw, but the other 17 being remote code execution flaws. And that's, again, 17 remote code execution flaws in their DNS server coming to light now. We don't know how long they've been there. It's just astonishing to me.

Anyway, I mentioned that Yuki was well known, and apparently especially so to Microsoft. Last summer he tweeted: "Made #1 this year." He said: "Thanks to @msftsecresponse and the bounty team. Congrats to all researchers on the list." He said: "I personally known some researchers who reported nice bugs but are not on the list this

year. Hat tip to their great work, too." So that was gracious of Yuki. And we haven't shown this for a while. We did several times before. This is MSRC's, the Microsoft Security Response Center, 2021, basically their MVP, their Most Valuable Security Researchers list. In this case it's the top 58. Yuki is number one. And then there's just 57 others.

So anyway, I think it's cool that Microsoft is giving them recognition. There was an interesting story that didn't have enough meat in it to make the podcast, but it was that Microsoft had increased the bounty for some of their products, some of their 365 things. So anyway, that's Patch Tuesday. And we will be getting to this most worrisome of all RCE in the remote procedure call runtime library by the end of the show.

Okay. So WordPress once again is in the crosshairs. And it's not surprising. There are just a constant barrage of problems with their plugins, and WordPress is so highly used on the Internet that the vulnerabilities matter there. A site and a service named Plugin Vulnerabilities - I don't think that was wisely named. If I were to tell my trademark guy that I want to call a site Plugin Vulnerabilities, he would say, no, that's a bad idea. You probably can't actually trademark that because it's just not unique enough.

Anyway, they explain themselves as "a service to protect your site against vulnerabilities in WordPress plugins." And as we'll see, not only will that keep them busy, but what's happened is we've seen sort of a sub-industry get created from third-party services that are all jumping in to perform that function. This Plugin Vulnerabilities site and service most recent posting from last week is titled "5+ Million Installed WordPress Plugin Elementor Contains Authenticated Remote Code Execution (RCE) Vulnerability."

They wrote: "Late last week, third-party data we monitor showed what was possibly a hacker probing for usage of a WordPress plugin named Elementor, which has more than five million active installs according to WordPress." And this hacker was probing for the file /wp-content/plugins/elementor/readme.txt. So that makes sense. If that directory were world-readable, and I guess it sounds like it is, they thought that they were going to get it, that would allow somebody to pull the contents of readme.txt. If they were able to do that, that would tell them that that site had the Elementor plugin installed, and then they would do whatever it was they were going to do.

The guys at Plugin Vulnerabilities said: "We couldn't find any recent disclosed vulnerabilities that should explain that, so we started doing our standard checks we do in a situation where a hacker may be exploiting an unfixed vulnerability in a plugin. What we immediately found was that the plugin isn't handling basic security correctly" - shock, I know - "as we found many functionalities where capabilities checks were missing where they shouldn't be. While some of those were not accessible to users that shouldn't have access, we found at least one that is, and the functionality accessible leads to one of the most serious types of vulnerabilities, remote code execution. That means that malicious code provided by the attacker can be run by the website."

They wrote: "In this instance, it is possible that the vulnerability might be exploitable by someone not logged into WordPress, but it can easily be exploited by anyone logged into WordPress who has access to the WordPress admin dashboard. Unless another plugin restricts access to the admin dashboard, that would mean anyone logged into WordPress would have access. The vulnerability was introduced in the plugin in version 3.6.0, which was released on March 22nd." So that's interesting. That's a little over a month ago, or rather just about exactly a month ago.

And they said: "According to WordPress's latest stats, 30.3% of users of the Elementor plugin are now on version 3.6.something." So that's an interesting stat all by itself. So that says that a month after the 3.6.0 was published, one month later only 30.3% of users of that plugin were using 3.6.something meaning that, first of all, 70% that had

not upgraded had avoided vulnerability as a consequence of that. But unfortunately in this case, because a vulnerability was introduced with 3.6.0, and we don't know when it was removed, then - actually we're going to find out in a minute. But 30% of users were vulnerable. And apparently some bad guy perhaps knew that and was poking around the Internet trying to find victims.

Anyway, they conclude: "Based on what we saw in our very limited checking, we would recommend not using this plugin until it has had a thorough security review" - meaning that overall they were not impressed with the design of the plugin that they found - "and all issues are addressed. That it has five-plus million installs and hasn't been properly secured should be very concerning," they wrote. "It certainly isn't for a lack of money at the developer, as they raised $15 million in 2020. It also isn't for a lack of reason to be concerned, as two years ago it was claimed a zero-day vulnerability in the paid version of the plugin was being exploited."

So their English is a little spotty, but they appear to be a legitimate security concern. Although, having said that, as I dug more deeply into this, I became a little bit, I guess, their background became a little more questionable. They posted snippets of code which demonstrated and detailed the vulnerability. And apparently this disclosure, or their disclosure, was made deliberately and irresponsibly over a dispute with the WordPress plugin forum moderators. It seems that these Plugin Vulnerabilities guys have long been unhappy with the way WordPress manages the reporting of security vulnerabilities. And reading between the lines, it sounds as though WordPress downplays, in their opinion, WordPress downplays vulnerability reports, which annoys these guys. And it sounds like there's been sort of a back-and-forth clash of egos.

A different firm known as Patchstack, in the same business as the Plugin Vulnerability guys, that is, another one of these companies in the add-on WordPress "we're going to try to keep you safe" business, wrote, they said: "The widely popular WordPress website builder plugin Elementor, which has over five million active installations, has recently released version 3.6.3, which contains an important security fix." So that tells us that .0, .1, and .2 were not fixed; .3 was. They said: "This vulnerability allows an authenticated user, regardless of their authorization, to upload arbitrary files to the site. The arbitrary file upload vulnerability could allow someone to take over the entire site or perform remote code execution. Please update immediately!"

Then they added that: "Patchstack Pro and Business users" - and this is them promoting their service. "Patchstack Pro and Business users have received a virtual patch to be protected from this vulnerability." And the previous group, the Plugin Vulnerabilities guys, who have their bruised egos, also produce what they call a WordPress Firewall to autonomously protect their subscribers from this danger when it's been configured to do so, in this case to limit the types of files which can be uploaded because it's by allowing you to upload something that you're then able to execute that, the attacker is able to cause it to get executed, thus running their code on your site.

So WordPress's undeniable vulnerability, I mean undeniable popularity, excuse me, their undeniable popularity, coupled with the constant stream of problems mostly created by insecure and poorly written WordPress plugins, has, as I said, spawned an industry of add-on WordPress protectors. In the past we've often referred to the firm Wordfence, where a lot of these vulnerabilities were found and reported, always responsibly. Those are good guys, and their business is protecting the sites that are their subscribers until the add-in has been patched. So they're as proactive as they can be.

And of course we just talked about Plugin Vulnerabilities and Patchstack. There are many others. I saw a list of about 30 recently. And although it was obviously self-serving, recall that it was Patchstack who, and we cited them a month ago, released a research whitepaper observing that last year saw a 150% increase in reported WordPress

vulnerabilities compared to the previous year, 2020, with the alarming news that 29% of the critical flaws in WordPress never received a security update. You know, they're plugins that their developers have wandered away from, yet later a critical flaw is found, and it never gets fixed.

They also observed comfortingly that only 0.58%, so, what, about one in 200 problems, were found in the WordPress core, with the rest being in themes and plugins written by anyone else and typically offered without review. They also noted that almost all of the problems, 91.38% of the flaws, were found in free plugins; whereas paid/premium WordPress add-ons only accounted for 8.62% of the total. So the takeaway, walk away advice is stick with the most barebones WordPress installation possible because the WordPress core is solid.

If you want more, take your time, look around, find the most reputable source of WordPress add-ons that you can, which probably means you have to pay for it. But you're paying for some security that's probably worthwhile. And in making this tradeoff accept that, as always, there will be a tradeoff between security and ease of use. You just need to decide where you want your site to sit along that tradeoff.

When we hear the phrase "Apache Struts," certainly long-time listeners to this podcast will immediately think back to the historic Equifax breach of 2017, which was the result of Equifax not having updated their Apache Struts Java web framework for several months after a critical "must patch" update had been published. Wired Magazine's coverage of the event at the time was titled "Equifax Officially Has No Excuse," with the subtitle "A patch that would have prevented the devastating Equifax breach had been available for months." So as our listeners know, that infamous breach compromised the data of 143 million users as hackers exfiltrated the names, Social Security numbers, dates of birth, addresses, and in some cases driver's license numbers. In other words, everything you need to perpetrate identity theft, which of course was the big problem there.

What wasn't mentioned at the time was that the flaw was in an historically troubled Struts component, and actually not only Struts, but an historically troubled component known as OGNL. That's the Object-Graph Navigation Language. And guess what. It's a powerful interpreter and then some. OGNL is an open-source Expression Language for Java which simplifies the range of expressions used in the Java language. Among other things, OGNL enables coders to more easily manipulate arrays. But as it turns out, parsing OGNL expressions based on untrusted or raw user input has long been dangerous.

I got a kick out of what Wikipedia had to say. Listen to this, and don't try to understand all of it. Just sort of let it wash over you. Wikipedia said: "OGNL began as a way to map associations between front-end components and back-end objects using property names. As these associations gathered more features, Drew Davidson created Key-Value Coding Language (KVCL). Luke Blanshard then reimplemented KVCL using ANTLR" - which, you know, antler - "and started using the name OGNL. The technology was again reimplemented using the Java Compiler Compiler (JavaCC).

"OGNL uses Java reflection and introspection to address the Object Graph of the runtime application. This allows the program to change behavior based on the state of the object graph instead of relying on compile time settings. It also allows changes to the object graph." And finally: "Due to its ability to create or change executable code, OGNL is capable of introducing critical security flaws into any framework that uses it." Yeah. We'd like two, please.

"Multiple Apache Struts 2 versions," Wikipedia writes, "have been vulnerable to OGNL flaws. As of October 2017, the recommended version of Struts 2 is 2.5.13. Users are

urged to upgrade to the latest version, as older revisions have documented security vulnerabilities. For example, Struts 2 versions 2.3.5 through 2.3.31, and 2.5 through 2.5.10, allow remote attackers to execute arbitrary code. Atlassian Confluence has been affected by an OGNL security issue that allowed arbitrary code remote execution and required all users to update."

Okay. In other words, as we've seen before, not all ideas are good, and OGNL appears to be way too powerful and also particularly gifted at being bad. Wherever it's used it appears to wreak havoc. Back in 2020, a different OGNL Injection Bug, which is what all of these OGNL problems are called, was assigned CVE-2020-17530 with an attention-grabbing severity rating of 9.8. Back then, a pair of researchers responsibly reported to the Struts team their discovery of what they called a "double evaluation" flaw in Struts 2 versions 2.0.0 all the way through 2.5.25, so sweeping, basically from the beginning of Struts 2 to then, which could occur under certain circumstances.

The advisory for that CVE states that "Some of the tag's attributes could perform a double evaluation if a developer applied forced OGNL evaluation by using the %{...} syntax." And "Using forced OGNL evaluation on untrusted user input can lead to a remote code execution and security degradation." I guess you'd call it a security degradation if somebody's remote code was being executed on your system.

Okay. So finally, although Apache had resolved that known 2020 flaw back then, in the succeeding release, which was Struts 2.5.26 since everything before that had this problem, researcher Chris McCown later discovered that the applied fix was incomplete. And, you know, that's what you get when these things are just this complicated, like this is mind-bogglingly crazy. So he responsibly reported to Apache that the so-called "double evaluation" problem could still be reproduced in Struts version, which they thought they fixed, 2.5.26 and above, which resulted in the assignment of a recent CVE. Actually it was last year, but it's just been patched. It was CVE-2021-31805.

So today, users are advised to immediately upgrade to what is now current, Struts 2.5.30 or greater, and to avoid using forced OGNL evaluation in the tag's attributes based on untrusted user input because that's never going to turn out well. It's just not safe. And given what happened after Equifax apparently ignored similar advice back in 2017, I would be inclined to do whatever was necessary to stay current with Struts 2 if you or your organization or anybody you know or care about is using it. Be sure that you're running the latest, which is now 2.5.30, because my sense is this is exceedingly difficult to execute. This is not the low-hanging fruit that the script kiddies are going to use.

But if it was found that a site is using a vulnerable version of Struts 2, and based on the statistics we keep seeing of upgrade inertia it's doubtless that some are, it's then going to take somebody who really understands this stuff to engineer an exploit. But it's clear it's possible. And it's obviously even hard to fix this thing. So it's unfortunate that this OGNL is anywhere near Struts. But it sounds like it's very, very much more power than probably is necessary or that should be in there. But it is.

Okay. Are America's nuclear systems so old that they're unhackable? Which is an intriguing question. And I was put in mind of my meeting back in 1984 a dear friend of mine who was incredibly non-computer savvy. She was a realtor. We met when I purchased my home from her in 1984. She needed to access the realtors' MLS, the Multiple Listing Service, and she was using something at the time which she called her "modem." She just called it "the modem." It was actually, I was humored to see, a Texas Instruments Silent 700 thermal printing terminal, and remember with those original rubber cups on top that you would press the telephone's handset down into after first dialing into some computer.

Well, we became lifelong friends. And when the Internet happened, this MLS service moved online. So I set her up with a Windows 95 machine. Windows 95 was current at the time. Then many years went by with everything working fine until someone she had over to her home was shocked that she was still using Windows 95. I don't recall where the rest of the world was by then. Probably at least on XP. So she immediately phoned to ask why her computer had been allowed to become obsolete. And I said, "Judy, does it work?" Which gave her pause. She said, "Uh, yes." And I said, "Is there anything you need to do with it that it doesn't?" And she said, "No."

So I explained that there was a growing problem on the Internet with security. People were getting their computers hacked by clicking on the wrong thing. And she said, "Oh, yeah. That's happened to several of my friends." She said, "They got viruses in their computers that were sending out emails to all of their friends and infecting them, too." So I said, "Judy, did you receive those emails?" And she said, "Yes." And I asked, "And did you get infected, too?" And she said, "No." And I said, "That's right. And that's because your computer is too old to get infected by modern viruses. It uses an older and original sort of code which is in many ways better, especially for the few things you need your computer to do."

You know, basically she just used what she called "the Google," which she did not understand. She did not know that the Google was not the Internet because that was what she addressed. She confronted the Google. So she continued to use that machine happily for many more years until someone convinced her to buy a new one, after which she pretty quickly got herself infected. And I sort of felt a little responsible for that because I had never needed to lecture her on safe computing.

Leo: She had no antibodies.

Steve: Exactly.

Leo: She'd never been exposed.

Steve: I had exactly that thought, Leo. Exactly. Okay. So that brings us to a really interesting piece that ran in The Record last week that I wanted to share some parts from. It was intriguing to me, and I think it will be intriguing to our listeners, because it was interesting to see how much of the general philosophy of complexity and security, and the "if it's not broke, don't fix it" philosophy which we've developed on this podcast as a matter of self-preservation through the years, and how much of that was in quotes from this story.

So to set up their piece, The Record first establishes the context of the moment. They wrote: "As the Cold War drew to a close, a surprising contender emerged as the third largest nuclear power on earth: Ukraine. The country was home to some 5,000 nuclear weapons, placed there by Moscow when Ukraine was still part of the Soviet Union. Kyiv sent the weapons back to Russia in exchange for security guarantees from the U.S. and Britain and a promise from Moscow that it would respect Ukraine's sovereignty.

"Then President Vladimir Putin invaded in February. The nuclear option, which many thought had been largely removed from the table, was one of the first sabers Putin chose to rattle when he announced that Russian troops were moving into Ukraine in February. He reminded the world that not only did Russia possess nuclear weapons, but it was prepared to use them. Anyone who 'tries to stand in our way,' he said, will face consequences 'such as you have never seen in your entire history.'

"The threat raised an uncomfortable question. After decades of pursuing disarmament talks and assuming nuclear confrontation was a bridge too far, was the United States ready for the ultimate confrontation with Russia?" And of course nobody wants that. Okay. But so that's interesting. Amid all the talks of cyber offense and defense, I have found myself wondering, as I imagine our listeners may have, how strong our nuclear deterrent is in the face of reportedly quite capable foreign cyber attacking adversaries?

What I'm going to share from what The Record wrote, and which I've edited for the podcast, speaks to exactly that. They said, and again I've edited this to make it more understandable verbally: "Right up until three years ago, U.S. nuclear systems were using eight-inch floppy disks in a IBM System 1 computer first introduced in 1976. It was not connected to the Internet and required spare parts often sourced from eBay. Some analysts think America's slow walk toward modernization of its nuclear systems may turn out to have been a canny strategy. Because the systems are so old" - much like Judy's old Windows 95 machine - "they are practically unhackable."

Herb Lin, a professor at Stanford University and author of a new book titled "Cyber Threats and Nuclear Weapons," said: "There is a truism about computers, which is that when we have a computer, we always want it to do more." His book looks at the risks of cyberattacks across the entire nuclear enterprise. He says the "more" problem inevitably introduces vulnerability into systems, and defense officials have to think carefully about how to modernize. Amen. Please.

He said: "If you'll grant the point that the more you want a computer system to do, the more complex the system is that you have to build, then you take the second step, and you realize that complexity is the enemy of security." Yeah. Where have we heard that before?

Leo: I think it was Admiral Adama in "Battlestar Galactica," actually.

Steve: Yeah?

Leo: Remember, that's why Galactica survived the attack of the Cylons. He still had phones with wires on them coming out of the wall. They hadn't adopted the new technology, and they were the only ship in the fleet that survived.

Steve: Nice.

Leo: You see?

Steve: Couldn't be hacked.

Leo: Yup.

Steve: There it is.

Leo: Yup.

Steve: So Lin says that this is where things start to go wrong, much as happened with the Galactica fleet. He said: "Run the probabilities, and there's a chance that one of those many complex components could be vulnerable to a hack in a way no one had considered before. The cautionary tale is Stuxnet, the virus and worm that found its way into the Natanz uranium enrichment plant in Iran in 2009 and 2010. Stuxnet" - which appears in this writing to have been the brainchild of U.S. and Israeli intelligence services - "was able to take control of centrifuges used to enrich uranium gas inside the giant plant and, without anyone noticing, get them to spin so fast they broke.

"For a long time, the cause of the centrifuge failures was a complete mystery. Scientists were fired. Officials thought they were sleeping on the job or not maintaining the systems properly. It never occurred to anyone until much later that a cyber weapon could possibly find its way into a system that was air-gapped from the Internet and so closely watched. Stuxnet had probably been in their systems a year before they even discovered it." And of course this is a cautionary tale because it suggests that maybe sticking with eight-inch floppies was a good idea.

"The thinking has been," they write, "that America's geriatric nuclear weapons systems may actually provide an inoculation from this kind of attack." Lin wrote: "Many of the systems right now are so old that there's nobody, or few, very few people who know how to get at them. So right now the current assessment is that the nuclear command and control system anyway is mostly robust against a cyber threat." And of course that bar is high. When you're going to say we're robust against the nuclear command-and-control system being compromised, good.

Hayat Alvi, a professor at the U.S. Naval War College, studies these kinds of nuclear weapons issues. She spoke to The Record in her personal capacity. She says she has a mantra when it comes to our nuclear weapons systems, and I love it: "If it's not broken, it doesn't need to be fixed." Alvi says the calculus is pretty simple. "Why try to change something that has worked for decades? Assuming you change them to upgrade them to modern technology, you are actually inviting more risks and potential threats and sabotage into the system."

While officials have tinkered at the edges of the nuclear weapons systems, John Lauder, who used to direct the CIA's Nonproliferation Center, says: "Most of the systems we're using are from the '70s and '80s." And I'll just remind everybody that we've been staring at the chips on motherboards which all come from China, wondering and worrying whether that EEPROM might be more than it looks like, or whether the Ethernet connector could have a chip hidden inside it. So it's like just, please, leave everything alone.

He said there has been a general sense from people who worked in arms control that: "We had put together a set of agreements that would keep peace and stability." As a result, modernizing nuclear weapons systems seemed less important since there was a general sense that the weapons would eventually be phased out. "But Ukraine," he said, "was a wake-up call."

In 1979, about three years after the U.S. nuclear weapons program adopted that state-of-the-art at the time IBM Series/1 computer, William Perry, who was a top Pentagon official at the time, got a phone call. The voice on the other end identified himself as the watch officer. Perry, who would later go on to be Defense Secretary in the Clinton administration, recounted in his podcast At the Brink, he said: "The first thing the watch officer said to me was that his computers were showing 200 nuclear missiles on the way inbound from the Soviet Union to the United States."

Leo: Yikes.

Steve: "And for one horrifying moment," he said, "I believed we were about to witness the end of civilization." I mean, this actually happened.

As Perry weighed the possibilities, he concluded this had to be some kind of mistake. There was nothing going on in the world at the time that would have caused the Soviet Union to suddenly strike. Perry asked the watch commander to find out what had gone wrong with the systems so he could explain what happened to the President in the morning. It turns out someone had accidentally put a training tape into the computer instead of an operating one. As a result, what the computer saw was a simulation of an actual attack. It looked real because it was designed to look real.

Perry says that night fundamentally changed the way he thought about nuclear weapons. He came to the conclusion that simple human error could indeed lead to nuclear war. He said: "It has changed forever my way of thinking about nuclear weapons. Up until this, a false alarm, an attack by mistake, starting a nuclear war by mistake was a theoretical issue." Until it wasn't. So there have been many questions raised and interesting movies made surrounding the question of whether a human being would be able to follow an order, would choose to follow an order to turn the keys to launch a strike.

So understandably there's a huge urge on the part of those who are given the responsibility of protecting us to remove the human factor from the loop on the basis that it introduces a wildcard, an unknowable uncertainty that cannot be relied upon. The good news is that there are now so very many lessons which have since been learned about the true fragility of our supposedly advanced technology that it's at least reasonable to hope that ultimate control will not be centralized. And after all, all of the strength of the Internet arises from its inherently and brilliantly decentralized design. The good news is William Perry was in the loop, got the call, did like a reality check and said, wait a minute, it makes no sense that there's actually 200 inbound nukes from the Soviet Union. Let's double-check our systems.

So anyway, it's going to be interesting to see what happens. A little bit of I thought interesting insight that because our weapons just, you know, no one has believed we were going to probably ever need these things again. I'm sure they're being dusted. Oil is, you know, they're kept in functioning condition. But the technology has been pretty much left as it was in the '70s and '80s. And I say good because the only way, the only way it would be safe would be if we ourselves designed the chips and used our own foundries to make them and built systems that did it like from the sand on the beach up to working silicon in order to function. Everything else, everything in our systems these days we get offshore. And the world has changed too much recently.

Okay. A couple of bits of feedback from our listeners. Max Feinleib said: "Re the Coinbase segment." That's where I was talking about the unsatisfying login experience or trial login I had with Coinbase. I mentioned that last week. He said: "Gmail also has the same login process where you enter your email, and then your password on another screen." And so I thought, well, okay, they don't have to be the same screen. It's just that entering your username, your email, has to not provide any feedback.

Okay. Many other listeners have written since then to note that this or that online service does the same thing as Coinbase. I distinctly recall, Leo, that once upon a time we used Gmail as an example of this being done right.

Leo: Mm-hmm.

Steve: But sure enough, I tested Gmail. I went to Gmail using an incognito mode so it wouldn't have any knowledge of who I already was. And I entered the email "bingo-zonk-dingo-lingo@gmail.com," and I never had the chance to enter a login password. Gmail immediately told me that they had no record of that account. So it appears that over the years ease-of-use has prevailed, even at Google, over stricter but less user-friendly and less forgiving login policies. And I guess, you know, it certainly is the case that not all websites should or need to set the bar at the same height. I guess I could see, although I don't like the idea of Gmail having broken this login process into something which can be decomposed into two separate things. What was it, it was not WEP, it was W, I want to say WPAD. It was that WiFi protocol where you had to enter in the eight-digit code.

Leo: WPS?

Steve: WPS, yes, WPS. An eight-digit code. Well, first of all, the final digit was a check digit for the other seven. So that didn't count. You always knew what that was. And then what we learned was that you didn't - the protocol was broken such that you could separately guess the first three digits, or maybe it was the first four, separately from the final three plus the check digit. And of course that hugely reduced the security, the fact that it was no longer all seven, because you could compute the eighth, all seven had to be right. Instead, you could decompose it into four and three, which reduced the strength by a factor of a thousand. Anyway, it does look, I just wanted to acknowledge, Max and everybody else who tweeted me and wrote saying Steve, you know, your favorite password manager does the same thing. And it's like, oh. Really? Okay. Well, that's too bad.

Leo: Why do they do that? I don't understand, besides the security issue, why ask for the email first? I've never - everybody seems to don't do that now. Is there some thinking?

Steve: Well, it is how they find your account; right? So that's how they identify you. But I would like them to be mum about it until they then ask you for your password.

Leo: Right.

Steve: And then say, we're sorry, we had a problem. Now, from a customer service standpoint, right, it's much easier if you tell the user, oh...

Leo: Oh, we don't know you, yeah, yeah.

Steve: Yeah. You have a typo in your email. Then the user goes, oh. But the problem is that's exactly that same ease of use makes it ease of hack.

Leo: Right. Yeah, because you can try email addresses till you find one that they do know.

Steve: Right.

Leo: And now you're halfway there, yeah.

Steve: And if it's, I mean, it's one thing if its Gmail. That's not good. But if it's Coinbase, or if it's your bank, or like something where security really matters, wow.

Leo: Now we can phish you, yeah.

Steve: Yes, exactly. Dana J. Dawson, he wrote, he said: "@SGgrc, on SN-866" - and today's 867, so that was last week - "you seemed unimpressed" - yeah - "with the new MS Windows Auto Update service." You think? "But in other SN episodes you've implored home router makers to include an auto update feature. Could you elaborate on this apparent contradiction on an upcoming SN episode?"

Yes. The issue, Dana and anyone else, the issue of auto updates is not settled because it's not black and white. There are pros and cons and tradeoffs, and there isn't a single right answer. The biggest difference arises due to the environment being addressed. For example, a complex enterprise network has many moving pieces, and it presumably has some dedicated IT staff who know those pieces intimately and are competent to test any changes and validate that updates are safe before they're applied throughout the entire network.

Contrast that to a cute little router box, sitting alone and forgotten in a closet, which was installed by an elderly couple's grandson after they asked him how they could get WiFi. If, when it was installed, that little router offered the option for automatically keeping itself up to date, we can hope that the grandson had the foresight to enable it if it wasn't the default.

My feeling is that given the "set and forget" environment which most routers find themselves in, coupled with the fact that those routers are bristling with features and run complex Linux operating systems where new problems are being found, and that there's active pressure to find and compromise these autonomous little devices for incorporation into botnets, cryptomining, and perhaps someday something more, I believe that the default setting should be to allow the router to repair itself in the event that a critical remotely exploitable flaw is discovered and reported to its manufacturer.

Yes, there should be an option to disable it. You should be able to say, and maybe even explicitly ask the user if you're nervous about having it on by default, this router will update itself only if it really needs to. Is that okay? Or maybe even have it be like three settings: off, all updates, or super ultra mission-critical updates only. I don't know. But the point is it feels to me like that's a case of the forgotten router in the closet, and Grandma and Grandpa don't even know what it is. It probably has socks covering it up now so it's overheating.

Okay. The enterprise and the router in the closet scenarios are obviously different. But what they have in common is their use of complex software which we still haven't figured out how to get right. I think that the use case of Autopatch, that is, Windows Autopatch, is the very large middle ground. I'm unconvinced that it will make sense for any significant enterprise that employs their own in-house IT staff to turn over control of updating to this level to Microsoft. But most small businesses do not have in-house IT staff. They've got some guy who comes in when they call to fix something that broke. If Microsoft has found that their monthly updating is breaking more things than it's fixing, then having a way to detect when something has gone wrong and revert, in other words Autopatch, makes sense. And "Autopatch" sounds more appealing than "Windows Un-Update."

So the problem is, to me, this whole thing, this Autopatch thing, and we're going to, you know, we've got four rings of testing. We're going to stick our toe in the water and see if something breaks, and if so we're going to back out, and we won't go any further. This all feels eerily like capitulation. As if somewhere inside Microsoft someone acknowledged that they cannot actually fix Windows, so they're going to automate it not being fixed. But, you know, I still love Windows. I think it's an amazing piece of work. It's in front of me right now. I think it's incredible that it works as well as it does, and I very rarely have any trouble with it. Although I don't know if you noticed, Leo, in the middle of this first sponsor break Zoom just shut down.

Leo: No, I didn't. Oh, how funny.

Steve: The window disappeared. And I just, like, stood there with my mouth open, what? That had never happened before.

Leo: Well, it's perfect timing.

Steve: It was. So I just restarted it and reentered our connection and...

Leo: There you are.

Steve: It was transparent. So okay. Every so often my cable modem does the same thing, and I just kind of shrug and restart it.

Leo: Computers.

Steve: I think it only happened once so far during - I know, Leo.

Leo: Don't understand how we survive, really.

Steve: I've been silent for the last couple weeks about SpinRite. But I thought I would share, like, what has been going on because it's kind of interesting. And I know our listeners who are waiting for 6.1 will find it interesting. Work has continued without interruption. We've just emerged from an interesting sideline where we learned some surprising things about many of our systems. One of the big changes being made in SpinRite 6.1 is the use, I've mentioned this many times, of very large 16MB transfer buffers. The theory was that this would allow maximum performance by reducing the overhead in between many little transfers, you know, intertransfer overhead.

And for SSDs this theory has panned out big-time. SpinRite is like two or three times faster than any BIOSes that anyone has. And often, as I did mention months ago, we're like right up at the ceiling of the maximum theoretical transfer rate on the wire. But this also holds for some spinning drives, which we now in the newsgroup refer to as "spinners." But it turns out that not all spinning drives perform best when they're handed huge transfers to perform. I don't recall why I thought to look, but we were finding that some BIOSes were outperforming SpinRite's own native drivers, which I have just written, which should not have been possible. So I added technology to benchmark

drives both ways, using SpinRite's new native hardware-level drivers and using the BIOS, with the intention to use the BIOS when it was faster than SpinRite because speed matters.

The BIOS has never been able to transfer in its history, still can't, more than 127 sectors at a time. That's just a fundamental limitation of the BIOS's API. But SpinRite can ask drives to transfer 32,768 sectors at once into a waiting 16MB buffer. You know, now we've got lots of RAM, so why not? Through experimentation, we discovered that some spinners would perform better when asked for 258 individual transfers of 127 sectors each, rather than a single transfer of 32,768 sectors. But my code was still not keeping up with the BIOS.

When I realized that the way I was breaking up the large transfer into smaller transfers was introducing 258 times the intertransfer overhead, I fixed that. I redesigned my drivers to be maximally efficient when large transfers needed to be divided into smaller pieces. And then, finally, SpinRite's own drivers were always at least as fast as the BIOS and often, especially for SSDs, up to several times faster. So SpinRite now never needs to use the BIOS when it's able to communicate with a drive's hardware directly. And SpinRite now measures the speed both ways of each drive using smaller 127-sector transfers and these mega 32,768-sector transfers, and remembers which one is best for that drive and adapter. So that's part of the drive enumeration process. It basically figures out how this drive wants to be talked to, and then that's what it does.

So as for SpinRite's development, I know all of this is taking time. The only thing I can say is that what is emerging is going to be a truly high-functioning piece of software, something that everyone will be proud to own and use. And also it's also creating a strong new foundation for SpinRite's future in v6.1.

Leo: Yay.

Steve: Yeah.

Leo: That's exciting. Speaking of RPCs and RCEs, Steve Gibson.

Steve: Yes, and you've right, Leo, about not letting things be exposed.

Leo: No.

Steve: This is going to be another big lesson along those lines. So it's CVE-2022-26809 with a CVSS, and we'll see why, of 9.8. Last Tuesday, when along with 127 other lesser problems Microsoft patched this flaw in Windows' remote procedure call runtime library DLL (rpcrt4.dll), it was unknown to anyone other than Microsoft and its independent discoverer, "BugHunter010," who is with the Chinese Kunlun Lab. He had privately and responsibly disclosed it to Microsoft. What we've learned is you can look at patches and reverse engineer them.

As the implications of this freshly patched flaw began to emerge, the entire security industry started sitting up straighter in their chairs because what we have here is something rare. In any not-yet-patched Internet-exposed Windows machines we have the basis for another zero-click Internet worm, reminiscent of 2003's MSBlast worm and the 2017 WannaCry attack. CERT/CC's Will Dormann tweeted that right now 1,329,075

Windows machines are publicly reachable and, until patched, are vulnerable. And all of those machines which remain unpatched, once the discovery is made of how to exploit this vulnerability, will be susceptible to full unauthenticated remote takeover.

Microsoft's own page disclosing and summarizing this vulnerability categorizes it as attack vector, network; attack complexity, low; privileges required, none; user interaction, none; confidentiality impact, high; integrity impact, high; availability, high; exploitation, more likely. In other words, Microsoft is quite aware that it doesn't get any worse than this. And at this very moment everyone, and I mean everyone, is working right now around the clock to figure out how to turn this into an active exploit. Security companies are trying to do it in order to build proactive defenses for their clients. They need to know what patterns to look for on the wire in order to block this. And those who know say this is going to be very difficult to block. We don't know what that means. We also know that all of the usual suspects in North Korea, in China, and Russia are burning the midnight oil to design packets that they can send to any listening server to take it over remotely.

We all know that when we start off with more than 1.3 million vulnerable Windows machines, one week ago, all of those machines were vulnerable. A month from now, somewhere between, what, 10% and 25% will still be vulnerable, and it will be their misfortune. A working exploit for this would be extremely valuable, and it would likely fetch a high price even though a patch already exists since everyone knows that merely reduces the size of, but does not eliminate, the target population. So if an exploit were developed it would be kept quiet since anyone selling it would want it to be rare, and anyone paying a high price would want to keep it for themselves or for their own selective resale. So I actually wouldn't expect to see a worm appear initially. Deploying this bad boy in targeted attacks makes a lot more sense.

Last Wednesday, the day after last week's patches became available, Akamai's security people, seeing how Microsoft had characterized this and seeing its 9.8 CVSS, became curious themselves about this change that Microsoft had made, and they posted a blog about the vulnerability because it's very likely going to become famous, or infamous. I've excerpted and edited from their blog to give everyone a sense for what Akamai did.

They wrote: "The CVE stated that the vulnerability lies within the Windows RPC runtime, which is implemented in a library named rpcrt4.dll. This runtime library is loaded into both client and server processes utilizing the RPC protocol for communication." They said: "We compared versions 10.0.22000.434 (unpatched from March 2022) to 10.0.22000.613 (just patched in April 2022) to produce a list of changes in various functions. The functions ProcessResponse and ProcessReceivedPDU caught our eye. The two functions are similar in nature. Both process RPC packets, but one runs on the client side and the other on the server side. We went on to 'diff'" - as in difference - "to 'diff' ProcessReceivedPDU, and noticed two code blocks that were added to the new version.

"In the patched code we saw that a new function was added after QUEUE::PutOnQueue. Zooming in on the new function and inspecting its code, we saw that it checks for integer overflows. Namely, the new function was added to verify that an integer variable remained within an expected value range after having another value added to it.

"Diving deeper into the vulnerable code in 'GetCoalescedBuffer'" - and, by the way, that is where the problem is - "we noticed that the integer overflow bug could lead to a heap buffer overflow, where data is copied onto a buffer that is too small to populate it. This in turn allows data to be written out of the buffer's bounds onto the heap. When exploited properly, this primitive could lead to remote code execution. A similar new call to check for integer overflow was added to three other functions, as well. The integer overflow vulnerability and the function that prevents it exists in both client-side and server-side execution flows." So that's what Akamai did. Basically they did what anyone would start

off doing, what everybody who is trying to figure out how to exploit that is doing right now. It's only been a week.

Marcus Hutchins, our friend, has posted a 7.25-minute video on YouTube titled "Exploiting Windows RPC - CVE-2022-26809 Explained | Patch Analysis," where he basically walks his viewers through a 7.25-minute tutorial on exactly what I just talked about Akamai doing. And the SANS Institute has posted an hour-long YouTube video titled "CVE-2022-26809 MS-RPC Vulnerability Analysis." In the video's description, SANS introduced - oh, and by the way, I have links to both of those in the show notes for anyone who's interested.

In the video's description, SANS introduced the topic by writing: "On Tuesday, April 12th, Microsoft released patches for CVE-2022-26809, reportedly a zero-click exploit targeting Microsoft RPC services. At the time of the publication of this abstract, there is no proof of concept available in the wild. However, based on the rating that exploitation is 'more likely,' we expect this won't last long." And of course I expect they're correct.

I've gotten this far into the subject without having made any mention of ports. Part of the reason is that RPC is not constrained to any single port, though it is typically carried over several, and one in particular. Believe it or not, the number one culprit is SMB port 445 which is used for Windows' infamous file and printer sharing. Everyone who's been listening to this podcast for any length of time will have heard me lament, many times depending upon how long you've been listening, that Microsoft's security for their Internet services has never, unfortunately, been worthy of trust. I mean, come on, their DNS server just now had 17 remote code execution flaws repaired. Just last week. The troubles with their remote desktop server have been numerous, and file and printer sharing has never been trustworthy. You cannot expose it to the Internet. Everybody's heard me saying that. Okay. Despite that, obviously many people have not heard me say that.

The research company Censys said 1,304,288 hosts are running the SMB protocol as of last Wednesday. Of those, 824,011, which is 63% of the total, were positively identified by Censys as running a Windows-based operating system, meaning that every one of those 824,011 machines was vulnerable last Tuesday. And Censys noted that they were unable to determine the OS behind approximately 28% hosts running SMB. So maybe some others were. They don't know. But they know that at least that 63%, 824,011 machines, were definitely running Windows. The majority of the systems running SMB are in the United States, with a count of 366,000 systems. And we've got a chart, a nice glow map of the world showing where they all are.

Leo: China is completely dark. But I don't think that's because they don't run SMB.

Steve: No. 366,000 systems in the U.S. Russia came in second place with a count of 144,622. And you can see it glowing there. Hong Kong had almost 73,000. Germany had just shy of 71,000, and France had 56,659. So this is a widespread problem.

Last Wednesday the 13th, CERT's Will Dormann tweeted: "CVE-2022-26809." He said: "Yes, blocking 445 at your network perimeter is necessary, but not sufficient to help prevent exploitation. If by April 2022 you STILL" - and he has it in all caps - "have SMB exposed to the broader Internet, you've got some soul searching to do. Now, about those hosts already inside your network...." And the day after that SANS tweeted: "Please remember port 445 is just one of the ports that may reach RPC on Windows. MSRPC does port 135, or in some cases HTTP as well." And actually it's a weird port on HTTP. It's 539 or something. I don't recall. And another researcher, Ned Pyle, tweeted: "CVE-2022-

26809 has nothing to do with SMB. It's an RPC vuln where a variety of transport can be used, like TCP/135 and SMB/445."

Okay. So no matter how much you want to, no matter how much you may need to, other than HTTP and HTTPS, and I suppose Exchange Server is necessary, too, Windows services and their ports must not be exposed to the public Internet. This is why site-to-site packet filtering by IP and port is crucial, and it's a perfect solution when IP endpoints are fixed, or even relatively fixed, and some other solutions such as a VPN, SSH, or port knocking needs to be used when a roaming IP must be able to obtain access.

So I just wanted to say that my choice at this point would be to run a WireGuard VPN from a server. We've talked about WireGuard before. It's a lovely, recently written, state of the art, 100% free and open source VPN which exchanges certificates for authentication, not passwords. So it is very strong. And unlike OpenVPN, it deliberately leaves the kitchen sink where it belongs, in the kitchen. It does just one thing and it does it well. And there are clients for all platforms now so connecting to it is no problem. Although it was originally built on Linux, Windows is now officially supported, and it's a single-click install.

So you could run a WireGuard VPN server on any Linux, Mac, or Windows machine on your network to which any number of roaming users could securely connect to obtain access to everything that you want them to on the inside. It's www.wireguard.com, and there's an install page that asks you which platform you want to download the installer for. There are now some "How to set up WireGuard VPN on Windows" pages. I've got them in the show notes also. But for those 1.3-plus million systems which CERT says were vulnerable last Tuesday, it's going to be interesting to see what we're talking about in the next few weeks to come. This thing really looks like, once somebody figures out how to take advantage of it, it's going to get loose.

Before we wrap up this topic and podcast, I need to mention what others have also mentioned, which is that the threat from this unpatched RPC RCE exploit is not only external. It's bad, yes, because it allows bad guys to get into your network. But there is no chance that bad guys who gain entry into an enterprise's network, once this thing has been weaponized, will not be adding a test for this patch being present to their lateral movement toolkits.

Remember that many initial network penetrations are with low and limited privilege. And the low and limited privilege is just the operating system trying to protect itself. And this is why privilege escalation or elevation exploits are almost as highly prized as remote code execution exploits. This RPC RCE will doubtless be added to all post-penetration kits because it would allow any unprivileged attacker who has obtained a minimal foothold inside a network to hugely expand their reach by moving laterally to other more valuable machines while simultaneously elevating their privileges on that destination machine probably to system level, which is where RPC typically runs. In other words, as those tweets were reminding, this is not just about preventing external public exposure to these services. The threat from internal abuse is very real, too.

And so we have last Tuesday an extremely rare, extremely worrisome problem that we just know there's no way that even in the U.S. that 866,000 machines are all going to be patched. And the race is on now to find a way to exploit this. Everybody is able to look at the code, reverse engineer the code, see what's gone on, and then work on figuring out how to send a packet in from the outside that will get some bad stuff done. In fact, given that it's been a week, unless it's impossible to leverage this, and it seems unlikely to be impossible, unless it's impossible it's probably been done, and we just don't know it publicly yet.

Leo: Well, there you have it. The inside story of the RPC RCE. M-O-U-S-E. Thank you, Steve, as always. This is why we look forward to Tuesdays. Tuesday is Security Now! day. We do the show at about 1:30, 2:00 p.m. Pacific, depending on when MacBreak Weekly is over. That would be 4:30 to 5:00 p.m. Eastern time, roughly 20:30 UTC. You can watch us do it live at live.twit.tv. You can of course, if you're watching live, chat live, either at irc.twit.tv or, if you're in Club TWiT if you're a member, you can do it inside the Discord. Always a lot of fun in there. We've been having some good conversations. In fact, we always do. That's where Stacey's Book Club is. Steve did an AMA. I think at some point we're going to do, we've got to do a Vitamin D episode in the Club. We have the Untitled Linux Show. We have the Giz Fiz. That's where we launched This Week in Space. We'll be launching, we'll announce it soon, another show in the Club. Because the Club members support it, we don't have to worry about when a show's starting out if we can get advertisers, that kind of thing. It really is a very important part of our operation these days, and we thank everybody who's a member.

You can get a copy of this show directly from Steve. He has actually two unique versions, the 16Kb audio, which sounds terrible, but it is small. He also has 64Kb audio. Sounds a little bit better, I must say. He also has transcripts, which is really great, so you can read along as you listen or search those transcripts for a part of this show or any of the 867 other shows. That's all at GRC.com.

While you're there, pick up SpinRite. Yes, I know, it's only version 6 right now, the world's best mass storage maintenance and recovery utility. But you will get a free copy of 6.1 when it comes out if you buy it now, and you'll get to participate in the development as we get closer and closer to the release of 6.1 and all its shiny new features.

Steve has lots of other free things there like ShieldsUP! and some great reading, all sorts of stuff: GRC.com. If you want to leave a message for Steve you could do it there, GRC.com/feedback. But probably the best way to reach out to Steve is through his Twitter account. He is @SGgrc on Twitter, and his DMs are open. So if you've got a Picture of the Week or a question or a comment or a suggestion, just DM him there: Twitter.com/SGgrc.

We have copies of the show at our website, of course, TWiT.tv/sn. There's a YouTube channel with all the videos. You can also subscribe; right? Sometimes that's the easiest thing to do. The RSS feed is available from that web page. Put that into your favorite podcast client, and then you'll automatically get Security Now! the minute it's available, which is a nice thing. You can, in some of those clients, review the show. And if you would leave us a five-star review, I'm sure Steve would just be glowing with happiness.