SolarBlizzard

Description: This week we open the New Year taking a longer look at fewer topics since the bad guys were apparently enjoying their New Year holiday, too. So we look at an interesting kludge that's been forced upon Chrome by ill-mannered antiviral scanners. We need to warn all enterprise users of Zyxel network border security products of another recently discovered built-in backdoor. We look at the rise in IoT compromise swatting attacks and a series of new flaws and vulnerabilities in the PHP Zend and Yii frameworks. We have a quick bit of miscellany to share, then I want to explain a lot about the value of trimming SSDs and newer SMR drives. And we'll conclude by catching up with what will hopefully be the last news, for a while at least, of the disastrous SolarWinds breach and intrusions.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-800.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-800-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with a defunct football; a very bad AV security practice that Chrome has had to do something maybe even worse to fix. We'll talk about the built-in password access in 100,000 Zyxel firewalls, VPN gateways, and access point controllers; an update on SpinRite 6.1; and a whole lot more. Security Now! is coming up next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 800, recorded Tuesday, January 5th, 2021: SolarBlizzard.

It's time for Security Now!, the first show of 2021, Show 800, 800 episodes. Steve Gibson, GRC.com. Congratulations.

Steve Gibson: I love that, 800 today.

Leo: Yeah, wow, wow. Well done. Feels like 800 years.

Steve: Yeah, well, it makes the math easy to do, Leo, because we know now that we have 199, counting this one, to go.

Leo: True, true.

Steve: Before I run out of three digits, and then we're going to have a real dilemma on our hands.

Leo: Well, we do, what, about a hundred - no, no, 50 a year, 52 a year. So we've got time.

Steve: Yeah, we've got four years. We've got four years.

Leo: We've got four years, yeah. There you go.

Steve: So this week we're going to open the new year taking a longer look at fewer topics since the bad guys were apparently enjoying their New Year's holiday as much as the rest of us were.

Leo: Good.

Steve: Not much happened, really. But there were some interesting things. We're going to take a look at an interesting kludge which I will remind our listeners is really one of my favorite words. Now that I know what "penultimate" means, we're going to all be talking about "kludge," which has been forced up on Chrome by ill-mannered AV scanners. We also need to warn all enterprise users of Zyxel's network border security products, in which have been recently discovered another built-in backdoor, which is just unconscionable. We're going to look at the rise in IoT compromise swatting attacks. Really, this one's going to bone-chill you, Leo, because I know you, and you're just going to...

Leo: Uh-oh.

Steve: And Lisa will be vindicated in her decision not to have any cameras inside the house.

Leo: Oh, geez.

Steve: And also we have a series of new flaws and vulnerabilities in the PHP Zend and Yii, I guess that's how you pronounce it, Y-I-I frameworks. We have a quick bit of miscellany to share. And then I want to explain a lot about the value of trimming SSDs and newer SMR drives as a consequence of something which has just come up. And I do have something of an announcement. And then we're going to conclude by catching up with what will hopefully be the last news, maybe for a while at least, of the disastrous SolarWinds breach and intrusions, which is the reason that this podcast is titled "SolarBlizzard."

Leo: It's more than wind.

Steve: And we do have kind of a blast from the past Picture of the Week. So I think another fun podcast for our listeners.

Leo: Good, good. First one of the new year. Did you have a good New Year's? Did you do anything fun?

Steve: Yeah. No. Lorrie has some pretty bad asthma. So she's already a little bit respiratory compromised. And so we're double extra careful about exposing ourselves to any environment where COVID could happen. She had been doing grocery shopping early in the morning. Like when I would get up and head off, as I sometimes do, at 5:30, she'd take that opportunity to go be the first person at the grocery store, under the theory that there were just fewer COVID germs floating around in the air. But now we've taken to using InstaCart and just having somebody else go pick up things.

Leo: I think it's better, yeah. Especially because you're in the L.A. area. All the ICU beds are full.

Steve: Yeah. And then she, like, wipes down everything. I mean, even though somebody else picked it up. And things that are delivered to the front door, they sit for a few days.

Leo: Yeah, let them age.

Steve: Unless, I mean, so...

Leo: Good, I'm glad you're taking care of yourselves.

Steve: So no, so we really are - and the fact that, at this point, if we only wait a few more months...

Leo: So close.

Steve: Because we're in our mid-60s, we'll probably qualify, as the vaccine becomes available, to be vaccinated. And once we're on the other side of that and have measured our antibody response to verify that it worked, then the game changes.

Leo: Party.

Steve: I can't wait to get out of the house and go support all my favorite restaurants because I miss that.

Leo: We were talking on TWiT, you know, the last time this happened, 1918, the Roaring Twenties happened after it was over. I think we're going to have a Roaring Twenties here, too. I hope so. I look forward to it.

Steve: Well, and it is the case that restaurants are one of the most impacted things.

Leo: I hope they survive, yeah.

Steve: And they never, I mean, restaurants are running on a razor-thin margin as it is, yes.

Leo: Anyway, yeah. We've had a lot of closings here in Petaluma. There's just a lot of empty storefronts all over town now.

Steve: Yeah. In the future they'll recover. I don't even know if my favorite places are still around. I'll go visit them after I've been vaccinated.

Leo: All right, Steve. Shall I show everybody this amazing Picture of the Week?

Steve: And you know, Leo, while you were talking, I was thinking, you know, maybe this was not time-based. This might have been event-based. I don't remember now. Because I remember worrying about if I ever needed to change the battery, if I opened it up, would it lose its code.

Leo: Oh, would it lose - yeah.

Steve: But maybe, I don't remember if I was worried about it losing its time or losing the key, the cryptographic key.

Leo: But you have to press a button on this; right? So that's why, you might be right, it might be button press; right?

Steve: Yeah. Although that also could have just been to energize the display, so they were wanting to keep the power consumption down. Anyway, we were talking cryptically about what we used to call the "PayPal Football."

Leo: My first dongle.

Steve: Yeah. And it looks to me from the display, I mean, it's been so long I haven't looked at it or thought about it. But it looks like it's eight digits, for one thing. So this picture was tweeted from one of our listeners to me with a little tear in his eye. Rather than showing the apparently eight-digit code, it shows "batt 05." And maybe that's the battery level, or I think it's just an error code saying battery is low, you'd better get this thing replaced before long. Mine, I still have mine. There it is.

Leo: I'm sure it's going to be somewhere, yeah.

Steve: It's completely off. I press it, and it doesn't even bother saying...

Leo: This one was for PayPal. But it was my first authenticator. You know, shortly thereafter they added other ways of doing it, including text messaging and so forth.

Steve: Right. Well, so actually this was from VeriSign. And so it was privately labeled by PayPal, but it was actually produced by VeriSign, and remember that they also had some eInk credit cards.

Leo: Yes, yes.

Steve: And those were definitely event-based because they really had to be lean on power consumption. But the point is, even this may have been event-based because, when you gave the number, I think one of our sharp listeners, as I recall, noticed that the least significant digit incremented. Every time you pressed it, it looked like you got random gibberish, but when someone pointed it out, it was like, I'll be darned, he's right.

Leo: Well, that's not good.

Steve: Like the least significant - well, and the point was that helped it deal with, I think, deal with time-based drift because it would - so you were only losing one digit of entropy by having the least significant digit increment. Still, you had seven, which was even more than today's six because, you know, the standard that we settled on is a six-digit PIN which changes every 30 seconds. So anyway, I just thought this was kind of cool. I mean, back in the early days of the podcast, this was the first attempt at a multifactor authentication scheme.

Leo: That's right. That's how long we've been doing this. This predates authenticators and all. Now, an authenticator is a time-based one-time password, so it hashes the current time of day with a secret number to get that six digits. How would event-based work?

Steve: Well, Yubico's are not time-based. They are event-based.

Leo: They don't have a clock; right. Yeah.

Steve: The YubiKey, yes, it changes every time you do it. So in the time-based, they're using a clock to drive the cipher that produces the digits.

Leo: And I could see how that would work because the site you're on in the app also knows the time, and they could then compare it. But how would they know what would be the right number if it's event-based?

Steve: And that's the problem. So in the Yubico it's a counter which increments every time you press it. So the idea is that there the individual sites are not empowered to

determine the code. They forward the code to Yubico that keeps the synchronized counter in its server; and it says, yup, that's a valid code for this person. And then upon receiving it, they then increment it on their end. So basically your device and a central server are maintaining a synchronized count by virtue of you from time to time sending them the most recent display from your device.

Leo: Oh, that's interesting. I had no idea. And the YubiKey code is very, very long. Some of it's static. The most significant stuff is characters, is key-specific; right?

Steve: And that identifies you to the server so that it knows which database entry to go look at.

Leo: Ah, of course; right.

Steve: Instead of needing to, like, look at them all and go, wow, who is this? We don't recognize this person.

Leo: Right, right. That's very - I never even thought about that. I thought, well, must have a clock in there. All right. Clever.

Steve: Yeah, cool. So, okay. We appear, and we'll touch on this in several different instances today, to be having more and more problems with the collision of evil forces with the forces of good, which are of course working to thwart the forces of evil efforts. We've seen, for example, how it's no longer sufficient for code to simply be signed with a certificate, like that solves the problem. Because today, as I experienced when I got a new certificate, newly minted code-signing certificates are not initially trusted, no matter the reputation of the certificate's signer or signee.

After all, since fraudulent certificates are possible, the only safe presumption must be that anything new that hasn't had time to earn itself a reputation must be mistrusted by default, even if it means generating some worrisome false positive alarms for users, until that reputation has been obtained or earned. And even after going through all the trouble of obtaining a certificate and waiting while it earns a reputation for itself, the recent SolarWinds Orion experience has just demonstrated for the world that even code signed with a long-trusted certificate can still be evil. So, yeah. Everything is a mess.

Just yesterday I received a report in the GRC forums that McAfee AV was declaring one of ReadSpeed's DOS files, Splash.com, to be a trojan. Okay. Splash.com is a splash screen program that I wrote for ReadSpeed. It is literally a splash screen. When it's run, it puts the VGA adapter into 16-color 640x480 graphics mode. It sets the color palette to the optimal 16 colors for the image. Then it decompresses the image, whose data is the rest of the program, into the VGA's screen memory. Since most of the screen is black, with only little blips of nonblack, I wrote a simple run-length compressor for that purpose. And Splash.com contains a matching run-length decompressor. So it must be that the random binary noise that's contained within the body of the program, by the purest of coincidences, matches some heuristic signature, causing McAfee to believe that Splash.com, a DOS program, is a trojan.

So last night, out of curiosity, while I was assembling this podcast, I decided to introduce Splash.com, which I just recently assembled from source code myself, I introduced it to VirusTotal to see how widespread this false positive was. I put a screenshot of the top of

VirusTotal's display in the show notes. So it shows two out of 60 engines detected this file.

And as you can see, the scan is dated 1/05/2021, so that's in UTC, so that was last night. The size is 15.2K, and exactly two engines said, "Whoa, Danger, Will Robinson. You have a DOS trojan." And that's McAfee that thought it was the Waft, whatever that is. Daft is more likely. And then McAfee-GW-Edition. So yes, two McAfee engines, nobody else in the known universe, two out of 60 thought this was a problem. And yeah, you know, if this actually was a trojan, you'd have to feel kind of sorry for it, Leo, since it would be running in...

Leo: In DOS. What are you going to find?

Steve: ...16-bit DOS, with hardly any memory around, and no Windows OS API to attempt to subvert.

Leo: Can it read FAT-16?

Steve: No privilege to elevate to. No return-oriented code to jump to the end of. No TCP/IP stack. The Internet would just be a far-off dream.

Leo: Was it a TSR?

Steve: No. No, it's not even - doesn't even try to stay resident. The moment you - you either hit Escape or Enter, depending upon whether you want to go into the Benchmark or go to DOS prompt. So it would be stranded and cut off from the world, sitting there, like, twiddling its little trojan thumbs.

Leo: Apparently SpinRite will also trigger VirusTotal. There's something about your assembly code, dude.

Steve: No one's ever seen anything like it. Huh. This is suspicious. Nobody writes assembler any longer.

Leo: It's assembly? Holy cow.

Steve: Only crazy hackers do that.

Leo: Yeah, Dave Redekop ran it, and three engines think SpinRite is up to no good. That's hysterical.

Steve: Well, actually there is an interesting back story to that. Presumably - now, how big is that? Is it like a hundred and something K?

Leo: 169K, yeah.

Steve: Okay. So that's the Windows EXE, which is not signed because, when I wrote the ecommerce system, back when SpinRite 5 was current in the early 2000s, digital signatures hadn't happened yet. And the way SpinRite works at the moment is that every licensed version has the user's name and the serial number burned into the copy of the EXE.

Leo: Oh, that's clever. That's a good idea.

Steve: Yeah, so they're all getting their own custom EXE. What that would have meant is that I would have had to do on-the-fly signing. And as a matter of fact, I was just researching how to do that.

Leo: Oh, yeah, because the signature's different for every one; right.

Steve: Yes. And so the fact that it is, as David's drop of this onto VirusTotal noted, since SpinRite is not signed, that raises arguably a flag. And what I was just researching the night before last was how to do automated EV signing of code because before long, happily, testers are going to want to be downloading SpinRite, and I'm going to want it to be signed, unlike SpinRite 6 that has just never been signed. Well, catch up with the 21st Century, Gibson. So we will be signing all future versions of this stuff.

Leo: Good, good, good, good.

Steve: Yeah. So anyway, all of that brings us to what inspired this preamble, which is a brand new instance of the cure often causing more trouble than the thing it's attempting to prevent. We have the news that AV programs, antiviral programs, have adopted the practice of preemptively locking newly appearing files out of an abundance, or perhaps an overabundance, of caution. They're doing this until that new file can be scanned and given the AOK for use by the system. So it turns out that under Windows 10 this new preemptive practice has been creating some trouble for Chrome.

And the cure for this involves, as I said before, one of my favorite terms: "kludge." Wikipedia defines "kludge" as: "A workaround or quick-and-dirty solution that is clumsy, inelegant, inefficient, difficult to extend, and hard to maintain." And Wikipedia said: "This term is used in diverse fields such as computer science, aerospace engineering" - where you really don't want a kludge; but, you know, duct tape - "Internet slang, evolutionary neuroscience" - where, yeah, pretty much humans are a kludge - "and government," where, yeah, that speaks for itself.

So what was Google's solution to antiviral scanners preemptively locking the files it was trying to work with? Retry the operation until it succeeds. Okay. This is so sad. It makes a mockery out of the term "computer science." There's no science here. This kludge was, you know, it's like, oh, we couldn't read the file? Let's try again. Couldn't read it? We still can't? Okay, let's try again. Wait, we still can't? Okay, let's try again. This kludge was recently placed into the Chromium commit-queue, and thus into Chromium, to address Bug 1099284.

The posting reads: "Retry ReplaceFile to protect against antivirus." And the text says: "Antivirus programs and other scanners may briefly lock new files which can lead to frequent problems with saving bookmarks and other files that use the ImportantFileWriter," which is the name of some function in Chromium. They said: "This attempts to deal with this by retrying the racy ReplaceFile step a few times." And they said: "This change also adds instrumentation to record how many retries are needed, for future tuning. It also moves the SetLastError call as close as possible to the function that will consume the last error code." And then they concluded: "This is done only on Windows because it's hoped to be the only place where it happens." So, you know...

Leo: I bet you they're right.

Steve: ...if you're catching the failure and retrying until the operation succeeds, it's not clear to me how maintaining a record of how many retries were needed helps. I'm sure they realize that this is a horrific solution. So perhaps adding some instrumentation makes it seem more highfalutin and covers up the fact that it's a horrible kludge. And of course Google is the victim here, not the culprit. From the standpoint of any program that's using the OS, the culprit is clearly the third-party afterthought add-on antivirus system which has effectively broken and significantly destabilized the underlying operating system.

So now all OS client programs like Chrome, Chrome happens to be the only one that we're seeing here who has preemptively hit this problem and is attempting to deal with it. Think about it. All client programs of Windows are, what, supposed to add retry code to their logic in order to deal with a badly written AV scanner that's causing the OS to transiently return errors? No, okay.

If the authors of these AV scanners are listening, and I hope they are, please consider this. Since any AV scanner that is doing this would need to hook the operating system's file system API, the proper way to do this is to suspend any OS client thread that's requesting access to any file that the AV system has not yet cleared for access. Then, once the file has been cleared, the thread that was attempting to access the blocked file would be allowed to continue. The request would succeed, and everyone would be happy. Chrome wouldn't be forced to loop on waiting for the file to be ready for it, which apparently it's a bookmark, so it just created. And apparently it then wants to open it in order to have an update of the bookmarks. And that's mysteriously failing because the AV has jumped in and locked it out from underneath it. That's just horrible.

So in that case the only thing that a client might notice if the AV were working correctly would be that the operating system call for the file took a bit longer than normal to complete. But only real-time operating systems provide guarantees of maximum call response time. And no one who has ever used Windows would confuse it with a real-time operating system. So again, whatever this AV thing is that's doing this, it has broken the operating system significantly with this behavior, by causing files which it's scanning to be locked and thus failing a call that should otherwise succeed.

And so this is one of these problems where the cure is worse than the thing that it's trying to prevent. There's nothing malicious about your bookmarks that Chrome just updated. And yet the AV system is causing Chrome to fail. What it should do is suspend any call which is trying to access a file that is currently being scanned, then decide one way or the other if it's okay or not and pass the call through, which keeps the OS from failing and allows the AV to be a good citizen in the OS, rather than causing these sorts of problems. But, you know, this is what we keep hitting on these days is where problems are being caused by the things that are supposed to protect us - which, Leo, is why you and I are both saying no to these third-party add-ons.

Leo: So did they ever indicate which AV is doing this? Is it more than one?

Steve: No, they didn't say. I imagine they know.

Leo: Oh, I'm sure they know.

Steve: And they probably don't want to, you know, yeah.

Leo: Some crappy AV probably, yeah.

Steve: Let's hope. Let's hope there aren't many of them. So I titled this one "zyfwp/" - that's all lower case, by the way. Then we have "PrOw!aN_fXp."

Leo: What is that?

Steve: If you just listen to this, you now have the ability to log into more than 100,000 Zyxel firewalls, VPN gateways, access point controllers on the Internet. It's unbelievable. Another, because this also happened in 2016, another hard-coded admin privilege backdoor account has been found that affects more than 100,000 Zyxel devices that can grant attackers right now, that are on the Internet, root access via either the SSH interface or the web admin panel. And as ZDNet put it in their coverage, and I agree, they said: "The backdoor account, discovered by a team of Dutch security researchers from Eye Control, is considered as bad as it gets in terms of vulnerabilities."

So owners of these devices, I mean, hopefully our listeners, if they have any Zyxel devices, by all means update. They are advised to update systems as soon as time permits and, in any event, to immediately, if not sooner, disable external access, if you can - and it's not clear to me that it's possible, for reasons I'll explain in a second - to the device's SSH and web admin access. Until that's done, anyone from DDoS botnet operators to state-sponsored hacking groups and ransomware gangs could abuse this backdoor, and I guarantee you they're at it right now, to access vulnerable devices and pivot to internal networks for additional attacks.

And here we are, 2021; right? No one can any longer be naive enough to imagine that these backdoors would exist, be publicly known, be fully documented now. I just read out the username and password because it's no secret, unfortunately. It's going to happen. So the device models affected include many of Zyxel's top products from its line of business-grade, not home devices as was the case in 2016, business-grade devices which are typically deployed throughout private enterprise and government networks: the Advanced Threat Protection (ATP) series, which is used primarily as a firewall, except apparently it has a now publicly known backdoor; the Unified Security Gateway (USG) series, typically used as a hybrid firewall and VPN gateway; the USG FLEX series, used also as a hybrid firewall and VPN gateway; the VPN series, which is obviously a VPN gateway; and the NXC series, which is used as a wide-area network access point controller, a WLAN.

So since the roles of these devices place them at the Internet-facing edge of a company's network, once they're compromised, attackers are able to pivot and launch attacks

against internal hosts. And again, after what we've been through in 2020, one should imagine that this is going on.

So at this time patches are available from Zyxel for four of the five device families: the ATP, USG, USG Flex, and the VPN series. Patches for the NXC series are expected this Friday, January 8th. In an interview with ZDNet last week, IoT security researcher Ankit Anubhav said that Zyxel should have learned its lesson from a previous incident that took place in 2016, our listeners may recall since we discussed it at the time. Back then it was tracked as CVE-2016-10401. Back then Zyxel devices contained a secret backdoor mechanism that allowed anyone to elevate any account on a Zyxel device to root, using the superuser password "zyad5001."

Ankit told ZDNet, he said: "It was surprising to see yet another hardcoded credential in Zyxel products especially since Zyxel is well aware that the last time this happened, it was abused by several botnets." He noted that the "zyad5001" password is still present in the arsenal of most password attack-based IoT botnets. They'll give it a go on the off chance that it may still work and give them root access.

But of course this time the situation is much worse. While the 2016 backdoor was a powerful elevation of privilege, it still required attackers to first gain access to a low-privileged account on a Zyxel device. So they could then elevate it to root. Today's backdoor grants attackers direct access to the device without any preconditions. And Ankit observed that, he said: "Unlike the previous exploit, which was used for Telnet access only, today's vulnerability requires even less expertise since it's possible to directly use the credentials on the port 443 web panel." And, finally, the targets will be far more interesting to higher end attackers. The 2016 flaw impacted home routers, whereas today's mess affects enterprise-grade devices.

The Dutch researcher who found the backdoor in his own Zyxel USG40 performed an - oh, and by the way, he simply dumped the firmware and looked at it. And there were the credentials sitting in the firmware, exposed for anyone to see. So this also begs the question, who else may already have found this and known about it, and for how long? The good news is a reputable researcher saw it, found it, verified it, and reported it responsibly to Zyxel. I have a link to Zyxel's vulnerability report where they titled it "Zyxel Security Advisory for Hardcoded Credential Vulnerability." And of course this can't be a surprise. They did it; right? It's in their firmware. So the story of why this happened is, in my mind, as interesting as the fact that it did.

So get a load of what Zyxel says in this credential vulnerability report. They said: "Zyxel has released a patch for the hardcoded credential vulnerability of firewalls and access point controllers recently reported by researchers from EYE in Netherlands. Users are advised to install the applicable firmware updates for optimal protection." Yeah. And they said: "What is a vulnerability? A hardcoded credential vulnerability was identified in the 'zyfwp' user account in some Zyxel firewalls and Access Point controllers. The account was designed to deliver automatic firmware updates to connected access points through FTP."

Okay. Then they said: "What versions are vulnerable? What should you do? After a thorough investigation" - which, you know, okay. They said: "...we've identified the vulnerable products" - good for them - "and are releasing firmware patches to address the issue, as shown in the table below. For optimal protection" - optimal protection, mind you - "we urge users to install the applicable updates." And they said: "For those not listed, they are not affected. Contact your local Zyxel support team if you require further assistance." Okay.

Leo: Well, at least it could be patched. That's good.

Steve: Yeah. So, you know, first of all, what is wrong with this picture? This doesn't really pass the smell test. They claim to have deliberately hardcoded, knowingly, right, remote admin access credentials throughout their enterprise-grade product line for the purpose of delivering automatic firmware updates to connected access points through FTP. Okay. So maybe they're just extremely inept, or perhaps they're not telling the truth. For one thing, the FTP port is not the one that's open. It's the HTTPS port which is open and authenticates on these login credentials.

So more than anything else, this looks very much like an on-demand backdoor access to more than 100,000 enterprises. And, you know, you wonder if Western users should worry that Zyxel is a Chinese network equipment manufacturer located in Taiwan. I don't know. But it just really is, I mean, you almost have to say that this could not have been deliberate because, if it were an attempt to put a deliberate backdoor into their products, they would have done a better job than just having the username and password in plaintext in the code. I guess actually the researcher did say that it was a hashed password, so he apparently ran the hash through a hash breaker in order to determine what it was. But the point is, it wasn't difficult for him to discover, and anybody else who was curious could have done the same thing.

And also my feeling is, if you want to know, if anyone wants to know how to do automatic updates, just look around. Update clients, like they're saying these enterprise-grade security products are, don't hold listening ports open, waiting for updates to be sent in. No. No one does that. Update clients periodically phone home to check in with the mothership to see whether anything has happened recently. That's what Windows does; right? That way no listening ports are ever externally exposed.

And the only thing a client needs to do upon receiving an update is to use a built-in public key to decrypt a hash of the file to verify its authenticity before copying it into its own firmware. Only the authorized publisher of the product's firmware would have the matching private key that's used to encrypt the updated firmware's hash and include it for verification of the firmware's authenticity; right? So this is easy to do. But instead, Zyxel is saying, well, we've got servers with open ports listening for incoming connections, and they will accept them if they authenticate with this username and password which is in our firmware.

I mean, if that's true, that's insanely wrong. And if they're saying this is the way we're handling updates is that we're going to be like listening for some sort of a FTP push update delivery system onto our clients, I mean, if that's the case, nobody should ever be using Zyxel equipment again. And in fact, just the fact that they seem unable to do this in a secure fashion should put anyone off of them. I used to like them. The hardware seemed solid and well constructed. I have a Zyxel dumb Internet switch around here somewhere, but I haven't used it for years because it was a 10/100.

But anyway, they have apparently failed to learn from their previous fiasco in 2016, and I'd stay as far away from these guys as I could. If you do own their equipment, by all means, go get an update. Apparently it's up to you to do so, despite the fact that they are claiming that their products auto update by receiving, you know - so, what? Does Zyxel have a list of all the IPs of all their equipment out on the public Internet, and calls them with updates? I don't know.

Leo: That wouldn't make sense at all.

Steve: Again, as I said, it does not pass the smell test.

Leo: A lot of times those backdoors are left in for in-factory testing or, I don't know, I mean - I don't know. This doesn't seem...

Steve: Yeah. The bad news is more than 100,000 of them are out there.

Leo: Doesn't seem nefarious. I mean, like what would be the point in doing that?

Steve: Yeah. Maybe it just, I mean, or maybe they don't want it to look nefarious, so they did a really clumsy job of it.

Leo: Just a bad job of it.

Steve: You know, say oh, well, obviously it wasn't on purpose.

Leo: Just to be clear, by the way.

Steve: Because look how stupid we are.

Leo: They're Taiwanese. They're not Chinese. So it's not like this is a Communist China plot to spy on us. They're Taiwanese. So, yeah, I don't know what the motivation would be. Who knows? It's a bad practice, obviously.

Steve: Yes. Not well-designed technology. Okay. So Leo, this is a little uncomfortable, but it's important. Swatting goes IoT. So as I'm sure everyone knows, the practice of swatting, you know, SWAT as in Special Weapons and Tactics, involves pranksters, malicious pranksters calling police to report a nonexistent emergency which, if it were real, would necessitate a forced entry, weapons drawn, heightened alert response from local law enforcement.

Of course, local law enforcement has no way of knowing whether any given call for help is real or a dangerous and expensive prank. So they might well be required to assume it's real, and that has certainly been the case in the past. Swatting attacks are like a real thing. Now, the U.S. Federal Bureau of Investigation, our FBI, is bringing awareness to an emerging trend where swatting victims' smart home security systems, and not just one or two, are being used to first launch and then observe swatting events. Last week the FBI posted a public service announcement titled "Recent Swatting Attacks Targeting Residents With Camera and Voice-Capable Smart Devices." I have a link to the public service announcement in the show notes.

The FBI wrote: "The Federal Bureau of Investigation is issuing this announcement to warn users of smart home devices with cameras and voice capabilities to use complex, unique passwords and enable two-factor authentication to help protect against 'swatting' attacks," their own word in their announcement. They said: "Smart home device manufacturers recently notified law enforcement that offenders have been using stolen email passwords to access smart devices with cameras and voice capabilities and carry out swatting attacks." Then they tell us what swatting is.

And they say: "Offenders often use spoofing technology to anonymize their own phone numbers to make it appear to first responders as if the emergency call is coming from the victim's phone number. This enhances their credibility when communicating with dispatchers."

So they said: "How is this version of swatting carried out? Recently, offenders have been using victims' smart devices, including video- and audio-capable home surveillance devices, to carry out swatting attacks. To gain access to the smart devices, offenders are likely taking advantage of customers who re-use their email passwords for their smart device. The offenders use stolen email passwords to log into the smart device and hijack features, including the livestream camera and the device's speakers. They then call emergency services to report a crime at the victims' residence. As law enforcement responds to the residence, the offender watches the livestream footage and engages with the responding police through the camera and speakers. In some cases, the offender also livestreams the incident on shared community online platforms.

"The FBI is working with private sector partners who manufacture smart devices to advise customers about the scheme and how to avoid being victimized. The FBI is also working to alert law enforcement first responders to this threat, so they may respond accordingly." And so they end this announcement explaining to use strong email passwords, practice strong cyber hygiene, complex passwords that you've not used elsewhere, don't reuse the password that you use for your email, use multifactor authentication and so forth. So bottom line, be really diligent about all of the good security practices we all know that we should be employing.

And so what we have here is another new way in which a failure to secure our perimeter might be used against us. And for what it's worth, this style of livestreaming of intrusion isn't new. Around Christmas a year ago, the publication Vice reported on a podcast called "NulledCast," which livestreamed to content-sharing platform Discord an incident where criminal actors hijacked a Nest and Ring smart home video and audio to harass the residents in creepy ways. And so I don't mean to creep everyone out, but this is what was shown. And I was thinking about, as I mentioned at the top of the show, Leo, how you had mentioned that Lisa didn't want cameras spread around the house.

Leo: No. Especially that drone.

Steve: Yeah, exactly. One incident captured a man talking to young children through the device in their bedroom, claiming to be Santa.

Leo: Oh, I remember that, yeah. That was terrible.

Steve: Yeah. Vice reported last year, and they said: "In a video obtained by WMC5 - Action News in Memphis, Tennessee - courtesy of the family, you can see what the hacker would have seen: a viewpoint that looms over the entire room from where the camera is installed in a far corner, looking down on the children's beds and dressers while they play. The hacker is heard playing the song 'Tiptoe Through the Tulips' through the device's speakers. And when one of the daughters, who is eight years old, stops and asks who's there, the hacker says, 'It's Santa. It's your best friend.'"

So Vice also reported finding posts on hacker forums offering simple Ring credential stuffing software - which as we now know, "credential stuffing" the term for brute forcing - as little as $6 for the software. And to their credit, Ring is responding. By February of 2020, Ring had rolled out an added layer of security beyond its already mandatory two-

factor authentication, which is the familiar one-time six-digit code to log on. They also now have alerts when someone logs onto the account, and tools to control access by third-party service providers which could also be breached. And Ring is also preparing to roll out, well, to roll out end-to-end video encryption, which was originally slated for release by the end of 2020. It'll probably happen soon.

Ring's announcement last September 24th said: "With end-to-end encryption, your videos will be encrypted on the Ring camera, and you will be the only one with the matching key stored on your mobile device that can decrypt and view your recordings." So we could argue that should have always been there, but this is the way we learn. And it is going to be there soon. So the takeaway is the more our digital technology becomes intertwined with our analog lives, the more inherent exposure our analog lives are going to be having to flaws in the digital domain.

So I guess I just, you know, it becomes increasingly important to really take security seriously. Clearly, people who are not security-conscious have a single password, probably for everything. And yes, they used it for their email, and their email account got compromised by some website breach somewhere. And they also use it to log into their Ring account. And so not surprisingly, bad guys figured this out and took advantage of it. Wow.

So we have a new serious problem in the PHP Zend framework. Many of our podcasts through the second half of 2020 contained mentions of potentially serious WordPress vulnerabilities; right? It got to be sort of a weekly event. It's like, okay, what's the latest disaster in WordPress add-ons? WordPress, of course, is written in PHP. And somewhat incredibly, where a server's implementation language can be determined from a scan, 79.1% of those servers are hosting sites implemented in PHP. So effectively 80%, rounding up, of the servers where the implementation can be determined, it's PHP. And well-known sites written in PHP include Facebook, 360.cn, Wikipedia, Zoom, Microsoft, VK.com, and of course WordPress. And two sites of note which recently switched to PHP are the Washington Post and Trip Advisor.

So PHP is clearly the current website implementation language of choice. And its usage is not declining. Its usage on the web has remained absolutely flat at 80% over the past year and a half. The runners-up, you know, the also-rans, are Active Server Pages, Ruby, Java, and Scala, which together consist of the remaining 20% of website implementation languages, with Active Server Pages having the lion's share of those. So it would be possible to simply write a site from scratch in PHP, and many people have. But over time, web authors noticed that they tended to be writing and rewriting the same things over and over because there are so many things that different websites have in common. So what this suggested was that PHP was still a little too low-level.

So frameworks were born to function as a sort of meta language function and class library implemented in and running on top of PHP. And these PHP frameworks provide classes which abstract many of the common ways which all sites work, and the things that all sites need to do. You know, like authenticating a user. Why write that login code again from scratch? Or storing site session data in a backend database. Why create that all over, every single time that you're being asked to create a website? So, yeah, obviously, lots of things that sites do are the same. And by creating a class framework structure, this allows site creators to focus more on the site and less on the redundant mechanics of implementing a site from a very low-level language.

So against this background, we have newly discovered trouble in one of PHP's more venerable frameworks, Zend. Zend is currently in use by more than 570 million installations. And it's the most used framework by enterprises. We've talked in the past about deserialization bugs in the context of Java, where a complex Java data structure needs to be stored or to be communicated. So it's a bit like compression and

decompression, and the decompression or deserializing phase is inherently an interpreter.

Unfortunately, as we've said, those who write the interpreters often implicitly assume that their serializer, which may always produce a sane and trustworthy serialization, is the only source, or will be the only source of the data that they are later being asked to deserialize. And that assumption often leads to security flaws. When attackers are able to provide the object to be deserialized, real trouble begins.

And sure enough, here we are again yesterday. An untrusted deserialization vulnerability was disclosed in the Zend framework, which currently has, as I said, 570 million installations. This flaw can be exploited by attackers to achieve remote code execution on PHP sites, maybe as many as 570 million of them. The vulnerability is being tracked as CVE-2021, the first CVE we're discussing in 2021, 3007. And from postings on GitHub, this also appears to impact Zend's successor project, known as Laminas, due to the fact that a significant amount of Zend's code, not surprisingly, was simply moved over into the new project's codebase. The vulnerability exists in the most recent, and essentially it's the last Zend framework, 3.0.0, because now Zend has become Laminas.

The good news is no exploits have been released so far. But then this just happened yesterday, so we'll stay tuned to see whether this ends up being turned into a weaponized attack. At this point, I looked, and I could not find - I found instances of this coverage of this vulnerability everywhere. No response yet from the PHP. It was in, unfortunately, in the "destroy" routine. And of course destroy is an often-called function. Anytime you have a dynamic language where you are instantiating and then later destroying objects, you're going to be calling that a lot.

And in another PHP-related, but unrelated to Zend, posting yesterday, another deserialization vulnerability was also discovered and disclosed by the German RedTeam Pentesting group. This one was found in the PHP framework Yii, Y-I-I. I have a link to their full disclosure in the show notes, for anyone who's interested. In their posting, the RedTeam guys note that insecure deserialization was #8 on OWASP's Top 10 list of web application security risks. And Check Point mentions presentations about PHP deserialization flaw exploitation dating as far back as 2010, so 11 years ago.

This isn't news. This isn't a new thing. But it's the generic problem of deserialization in PHP has long been recognized and is obviously still present today, 11 years after it began being talked about. And it's sitting there. I think this OWASP Top 10 had it on their list in 2017, and it's there today. So given PHP's crazy popularity, 80% of all websites, and the increasing intrusion pressure that we seem to be witnessing, I won't be surprised if we're talking more about deserialization flaws in PHP in the future. And Leo, now for a little extra pain.

Leo: Okay.

Steve: I will note that bitcoin is currently trading at $32,883.

Leo: You want to feel more pain? Goldman Sachs says they expect it to get to $146,000 in the next two or three years.

Steve: Oh, no kidding.

Leo: How many did you have, 50?

Steve: Well, okay. Now, the good news is one of your insights is the only thing that is providing me with solace.

Leo: What's that?

Steve: Because you commented that, had I not formatted and installed Windows on top of my bitcoin wallet, I would have traded them for dollars a long time ago.

Leo: Sure. Oh, yeah. When it hit a hundred bucks you would have traded it; right?

Steve: Yes, I would never - I would never. Well, we started feeling the pain when it originally touched on $20,000, like a while ago. That's where this happened and where it was like - in fact I think I did. Actually I went around and I made absolutely sure that I did not have a copy of my wallet anywhere. And I didn't. I'm absolutely sure of that.

Leo: Well, in some ways I feel happy that I can't unlock my wallet because I can't trade it in. So I'm just going to, you know, as it goes up, I watch it, and I think, I really need to unlock that wallet.

Steve: $100,000. That would be $5 million.

Leo: And then it starts to really hurt, yeah, yeah.

Steve: Oh, boy. It really does hurt, yeah. I have had some fun tweets from people saying, you know, back when you talked about it, I think it was worth, what, 50 cents for a coin?

Leo: If that. It wasn't even that.

Steve: And that's - right. And that's why, 50, oh, well, that was fun. Format.

Leo: Yeah, 25 bucks, big deal. No one had any idea. And that's why whatever I did with my wallet, I didn't, you know, I should have used LastPass and stored it. But it didn't matter. There was nothing in it. It wasn't worth anything at the time.

Steve: Well, and you know, the brilliance of it, we talked about it at the time, is that the curve of the rate of bitcoin production was preordained. It was going to plateau and level out. And it's been following the curve that Yakatomi or whatever his name was...

Leo: Satoshi Nakatoma.

Steve: That's actually - Nakatoma. Satoshi, just Satoshi, right.

Leo: Satoshi, yeah.

Steve: Yeah, so wow.

Leo: Nakamoto, yeah.

Steve: Nakamoto, right, yeah. I was joking because Nakatomi Towers was the building in "Die Hard."

Leo: That's right. That's right.

Steve: That was under attack.

Leo: So, yeah, it got to 34,000 over the weekend.

Steve: Oh, Leo.

Leo: It's a bubble. It's just a bubble. It's not, you know, it's got no real...

Steve: I don't think so. I don't think so.

Leo: Just keeps going up?

Steve: I think the fact that it is substantial and has become substantial, it has established itself as a storage of value, which it's, I mean, that's what it's going to be. There are currency exchanges. The IRS has woken up. They're going to want their piece of the action. And so, yeah, it's a real thing.

Leo: It's amazing.

Steve: So, ha. But again...

Leo: What a world. What a world.

Steve: I would never have known to hold them, so what the hell.

Leo: It's a cruel, cruel world.

Steve: Oh. And one of our listeners just sort of thought to correct me, and I thought, well, what the heck, I'll correct the record. I referred to Yahoo News's Kim Zetter as "he" last week. And it turns out "he" is a "she."

Leo: Ah.

Steve: So in my show notes I actually used Kim's name. But when I was just being a little more breezy, I said "he." But no. Sorry, Kim. Not that you care, but we all know you're a she now.

Okay. ReadSpeed and SpinRite. We are continuing to see some, well, actually a flood of very interesting results from our podcast listeners who are experimenting with ReadSpeed and their SSDs. So I have a link to a GRC forum post from last Wednesday by someone named Dale. And I've got the data here in the show notes, so we don't even really need to bring the post up. He wrote: "Hi. This WD 500GB Blue SSD was purchased 18 months ago and installed in a nine-year-old computer that served mainly, until recently, as a PLEX server, which probably accounts for the low speed in region 0."

And so he posts a run of the ReadSpeed Benchmark. And the Benchmark log puts a timestamp, so you can see when it was that it was taken. And so I got a kick out of the fact that this was at 12/28 at 16:19 was when he did this one. And so it shows the drive identity information. And what's really interesting is that for the first gig of that drive it shows that it was reading at 17.9 MBps, which is a snail crawl. 17.9 is lower than the inner cylinder of an old hard drive. The 25% is at 349.3. Then the 50% and 75% are at 544.7 each, exactly. And the 100% is at 541.4. So then he says in his posting, after running SpinRite at Level 3, and then we have, and the thing that I got a kick out of is that this one - okay.

So the first one is at 12/28 at 16:19. This one is at 12/30, meaning two days later, at 13:06. And so, yes, SpinRite currently probably did take two days to run on his half-a-terabyte WD SSD. But it did. It elevated the ReadSpeed of the beginning of that drive from 17.9 MBps to 326.5 MBps. It elevated the 25% point from 349.3 to 532.7. And then, interestingly, it dropped the 50% and 75% from their - each had 544.7, dropped them to 531.6 and 531.3, and the end to 534.5. So what does all that mean? Dale concluded...

Leo: Yeah, why would they drop?

Steve: Exactly. And that's why I want to talk about this. It's really interesting what's actually going on. He concluded by writing: "Nice, way to go, Steve," because the Level 3 scan hugely, from 17.9 to 326.5, bumped up. And similarly, the 25% point. So I very much appreciate Dale's enthusiasm for the fact that the read performance of the first gig of his 18-month-old half-a-terabyte Western Digital SSD jumped from 17.9 to 326.5. Obviously a huge improvement in his device's read performance. And it's very likely that much more than that first gigabyte of the drive was similarly improved. We don't know because currently we're only looking at the first gig. SpinRite'll do much more than that, of course. It'll do the whole drive. Given how slow the front of his drive was, it had to be that the drive was struggling significantly to read back its data.

So this huge recovery of performance must also signify an improvement in the drive's future reliability. At this point there is every reason to believe that this was a data-protecting thing that he did. But there is a tradeoff that anyone doing this needs to be aware of. And I wanted to talk about this so that everyone doesn't start running SpinRite

Level 3 on their SSDs without considering this. This tradeoff is apparent in exactly the question you asked, Leo, in Dale's 50% and 75% benchmark location numbers. His SSD's read performance in that region appears to have dropped from 544.7 MBps to 531 MBps. And I say "appears" because that's not actually what happened. Those regions of Dale's SSD were not slowed down by passing SpinRite over them.

Instead, Dale's SSD now believes that those regions are in service and containing valid data, rather than knowing them to be blank. And that is the important change. SSDs and recent SMR (Shingled Magnetic Recording) format spinning drives maintain a logical to physical mapping layer. When they are powered up, like for the first time, well, actually each time, this mapping is loaded into their onboard RAM for high-speed access. Logical sectors are what the user sees at the SSD's interface, and physical regions are where that data is actually stored. The creation of this separation, this abstraction layer, is the way SSDs perform wear leveling, by spreading repetitive writes to the same logical regions across different physical regions of the media.

Okay. Now, one might imagine that at the start, a fresh and empty SSD would have a one-to-one mapping layer with all of its logical sectors uniformly mapped out across the physical medium with, like, everything's just ready to go. But that's not the case. A brand new unused SSD has an empty mapping table because, if nothing has been written to the SSD, there's nothing to read back. So any read from an unused SSD doesn't even bother looking at the SSD's physical storage media. That mapping layer already knows it's empty. So in response to a read, the SSD instantly returns sectors of zeroes. And thus its read performance appears to be amazing. It saturates its interface's capacity.

And that's what those two 544.7s in Dale's 50 and 75% points were. They weren't the speed of his SSD. He had never written anything there ever. And so the SSD knew that those regions were empty, and it just sent back zeroes as fast as the SATA III interface would carry them. And that is 544.7 MBps. That's the theoretical maximum speed for a SATA III interface. Okay. But once a SpinRite Level 3 refresh pass was made over the entire drive, rather than being partially empty, now its mapping table is full, with every logical sector assigned to an actual physical region of the SSD's storage medium.

So after SpinRite's pass over the SSD, the ReadSpeed Benchmark revealed the true read performance of the drive's media. And it was quite a respectable 531.6 MBps. Not very much slower than a SATA III link's maximum speed. And we did observe that the frighteningly slow front region of the drive is now running far faster, and probably far more reliably for the future than it was going to before. But there's also a new problem, which SpinRite in the future will be fully aware of and will handle automatically, but which SpinRite of today does not.

And the easiest way to express what a current SpinRite Level 3 pass has done would be to say that SpinRite will have fully untrimmed the SSD by writing across its entire logical surface, thereby leading the SSD to believe that the entire SSD is in use. So leaving an SSD in an untrimmed state doesn't damage it in any way. But it can impede the performance of future writes to those regions. Okay. So in order to write to an SSD's media, you know, the actual physical medium, the electrostatic storage capacitors of its cells must first all be discharged and drained of their electrons. And only then can they be charged back up to the varying voltage levels required to represent the data bits which are being stored in each cell.

The hitch is that the erasing can only occur in relatively large page blocks which always contain multiple sectors, multiple logical sectors. So if the SSD knows that one of these page blocks already has all of its cells discharged and drained and empty, it's able to bypass the discharge phase and much more quickly write new data, a sector for example, into that empty region. All it has to do is charge up that sector's cells in a page that it already knows is blank. But if the SSD believes that all of the logical sectors within a

region may be containing valid data, because they have been previously written, if only by SpinRite performing a Level 3 pass, then in order to write one sector into that page, the entire page must first be read. Then the entire page must be discharged and erased. Then the entire page must be rewritten with only that one sector of its data changed.

And even when a lot of data is being written, not just one sector, writing one sector into a page is the worst case. Often you're writing all of a page's sectors. But even then, such that all of the sectors filling an entire page are being written, knowing that a page is already empty and discharged of all of its data allows the SSD to skip the discharge phase. It wouldn't bother reading it if it knew it was all going to be rewritten, but it would still need to discharge it all. But if it knows that the page is already empty, it skips the discharge phase and simply charges up the page's cells with their new data.

The process of informing an SSD that its logical sectors do not contain any useful data is known as "trimming." And all modern operating systems are now, today, trim-aware. And we've talked about this in the past. In fact, we talked about this not long ago when it was found that Windows 10 was attempting to trim its hard drives. It turns out that in that case this was being done in error by Windows 10. And it was generating kernel exception traces in its logs. But it turns out it's not as nuts as it might seem, since the latest SMR, as I mentioned, Shingled Magnetic Recording drives, actually share many of these same characteristics as SSDs. And they do support and can benefit from the trim command.

And in fact there are instances of the same behavior being reported by the ReadSpeed Benchmark where SMR drives are reporting impossibly high performance in some of their regions because they're not actually being read. The drive knows there's nothing there, so it just squirts zeroes back at 544.7 MBps. So a trim-aware operating system is continually updating the SSD or SMR drive with information when specific sector ranges no longer contain active file system data.

The most common example would be whenever a file is permanently deleted from the file system, rather than being moved into a trash container. When the SSD receives this trim information, it will likely unlink those storage pages from its sector mapping table and may place them onto a garbage collection list, which causes them to be scheduled for erasure in the background so that, when more storage is needed, already emptied pages will be ready to receive new writes.

And as I've mentioned before, SpinRite's awareness of its drive's contents will be raised from the drive's surface where it is today to include popular file systems. The various flavors of FAT and NTFS will probably be added first, followed by the various file systems supported by Linux and other important OSes. And this awareness will allow SpinRite to first of all run far more quickly on sparsely populated drives and to trim any empty regions that it might have the occasion to write to.

Now, the good news is today's operating systems often have the ability to trim on demand. For example, in Windows 10 the PowerShell command is Optimize-Volume space -DriveLetter space and then for example C: or whatever, space, then -ReTrim space -Verbose. And I have that down at the end of page 11 of the show notes. When you run that, you are telling Windows 10, please retrim this drive letter. Which is to say please go through the operating system and send trim commands to the SSD or SMR drive for all the known empty locations in the drive's file system. And the good news is also various Linuxes also have trim facilities which are often invoked periodically through CRON. So it may well be that Linux is just going to sort of catch up over time with any file system deletes that have been done.

So until we have a SpinRite that's aware of the file systems on our drives, and I'm doing nothing now other than working toward that goal, anyone who runs ReadSpeed and

discovers slow performance at the front of their drive, which postings into the GRC forum are revealing to be surprisingly common, and who then chooses to perform a Level 3 SpinRite scan to fix those regions, could stop SpinRite before it gets out into the nether regions of the drive which are unused, or let SpinRite refresh the entire drive, which might result in having the drive take defective regions it discovers out of service.

Then, after booting back into your operating system, ask your OS to manually perform a full trimming of the drive, which will release and unmap all of the drive's unused space. And after some period of time, the drive will have gone around and done its garbage collection, essentially preemptively erasing all the data from those now trimmed regions so that they are ready, standing by in a free page pool which the drive will automatically pull into service as it needs to do writes.

So there is everything we know about trimming. And, boy, it's so cool to see this actually happening using ReadSpeed and SpinRite. Oh, and of course, after you did the retrim command under Windows 10, you could rerun ReadSpeed again. And I imagine Dale probably will. I'd love to see his results. And what you should find is the end of the drive will be back up at that impossibly high performance of 544.7 MBps because essentially the retrim has again told the SSD, don't worry, there's not actually anything that you need to worry about out there, so you're free to return zeroes. Very cool stuff that we are seeing.

Oh, and the other thing you might try is just do a Level 2 first. We don't exactly know why, but what we're seeing is just performing the Level 2 scan is bringing a lot of the performance back. It's not really necessary to do a full refresh. And of course a Level 2 is only a read pass, unless SpinRite actually sees some problems, in which case then it will drop down into a higher level to actually do some repair work. So you might try a Level 2 first. It's also a lot faster than a Level 3 because it's just breezing along doing writes, and the Benchmark is demonstrating that a lot of the performance at the front of these drives is recovered after a Level 2.

And speaking of SpinRite, Leo, Sunday evening, two nights ago, I posted to my progress tracking blog over in the forums an entry titled "I am at work on SpinRite v6.1 code." I posted: "10 days after ReadSpeed's release, the shakedown test of the benchmark's new hardware drivers has been a tremendous success. We have identified only one problem with a system in Denmark using an old ASRock motherboard. I've located and purchased one of them in Germany, and it's on its way. So when it arrives, I'll see what needs to be done to get the drivers running on it, too. But aside from that single problem, as far as we know, this new technology in ReadSpeed is ready for SpinRite's use. So today, Sunday, January 3rd, 2021, I successfully assembled SpinRite's 31 assembly language source code files and created a working DOS executable. Now I begin the work of incorporating the new drivers into SpinRite."

And in fact the show notes, which you had onscreen there a second ago, Leo, show an image I snapped of Visual Studio's Solution Explorer panel showing SpinRite 6's file set, and the new project SpinRite 6.1 in the header there. So I am officially at work incorporating the drivers, which they have just turned out to be, well, as I said, surprisingly robust and ready to go. I mean, as far I'm concerned, as fast as I can get them into SpinRite, we're going to have 6.1.

Leo: How exciting. Very good news.

Steve: Okay. And our wrap-up with SolarBlizzard. A new flaw was discovered in SolarWinds Orion software. As we know, the way the bad guys were able to get their malicious Sunburst, as it's been called, malware deployed into victim networks was by

poisoning SolarWinds' source code repository so that all new builds of their - I've got the hiccups. It's not a Skype dropout, it's me hiccupping.

Leo: It's a diaphragmatic dropout.

Steve: Okay. Slow down, Steve. So that all new builds of their executables would automatically and surreptitiously include the attacker's trojan. Now we're learning the details of a different authentication bypass vulnerability which also exists in the Orion software, which may have been leveraged by adversaries as a zero-day - in other words, the bad guys, it may be different bad guys, knew about it - to deploy the other Supernova malware into specific targeted environments. According to an advisory published by CERT, the SolarWinds Orion API that's used to interface with all other Orion system monitoring and management products suffers from a security flaw, it's been designated 2020-10148, that could allow a remote attacker to execute unauthenticated API commands, thus resulting in a compromise of the SolarWinds instance.

The advisory states: "The authentication of the API can be bypassed by including specific parameters in the Request.PathInfo portion of a URI request to the API, which could allow an attacker to execute unauthenticated API commands. In particular, if an attacker appends a PathInfo parameter of 'WebResource.adx,' 'ScriptResource.adx,' 'i18n.ashx,' or 'Skipi18n' to a request to a SolarWinds Orion server, SolarWinds will set the SkipAuthorization flag, which allows the API request to be processed without requiring authentication."

And remember that SolarWinds' updated security advisory of December 24th made note of an unspecified vulnerability in the Orion platform that could be exploited to deploy rogue software such as Supernova. The details of the flaw had remained unknown until now. So this is probably what that was.

Basically, I mean, I hope this was a bug. I mean, it looks like of deliberate. But let's hope not. Let's hope that was just, I mean, the fact that you use the PathInfo, and you append Skipi18n, and that causes it to set the skip authorization flag, eh, okay. Again, Leo, as you said, it may be that this has fallen on hard times from a development standpoint. And as you also said of the Zyxel mess, sometimes lazy developers just put that stuff in so that they have access if they should ever need it. And but imagine being an enterprise customer and learning that, well, yeah, there's a backdoor in the product you purchased from these guys in case they ever need to get in. What?

Leo: Unh-unh. No. No. Eh.

Steve: And finally, last Thursday's Microsoft - and I had to double-take on this. They're calling it the Soroagate. I went, what? Sororogate. Sororagate?

Leo: Solorigate.

Steve: Yes, thank you, Solorigate.

Leo: That's not a good word.

Steve: So it was like, okay, Microsoft.

Leo: Not a good word.

Steve: Microsoft Internal Solorigate Investigation Update. It provided some interesting news, including that unauthorized access had been gained to some of Microsoft's source code. Okay. So that on its face sounds bad. But the posting also shares some interesting philosophy, which I thought our listeners would really find interesting.

Microsoft wrote: "As we previously reported, we detected malicious SolarWinds applications in our environment, which we isolated and removed. Having investigated further, we can now report that we have not found evidence of the common TTPs (Tools, Techniques, and Procedures) related to the abuse of forged SAML tokens against our corporate domains." You know, the forged authentication tokens was one of the things, one of the IOCs, the Indications of Compromise, which was found to be common and often abused in networks.

Anyway, they continue: "Our investigation has, however, revealed attempted activities beyond just the presence of malicious SolarWinds code in our environment. This activity has not put at risk the security of our services or any customer data, but we want to be transparent and share what we're learning as we combat what we believe is a very sophisticated nation-state actor. We detected unusual activity with a small number of internal accounts; and, upon review, we discovered one account that had been used to view source code in a number of source code repositories. The account did not have permissions to modify any code or engineering systems, and our investigation further confirmed no changes were made. These accounts were investigated and remediated."

And, now, here's the really interesting part: "At Microsoft, we have an 'inner source' approach, the use of open source software development best practices and an open source-like culture, to make source code viewable within Microsoft. This means we do not rely on the secrecy of source code for the security of products, and our threat models assume that attackers have knowledge of source code. So viewing source code is not tied to an elevation of risk."

And that was news to me. I think that's cool. So I wanted to share that. And it would be interesting to know whether this is the way Microsoft has always operated, or whether this philosophy represents some new enlightenment in the era of them being Linux-friendly and purchasing GitHub and all that.

Leo: Yeah.

Steve: Yeah. So that was very cool. And of course, as we know, that's the right way to design things. We've talked often about how the great step forward in the evolution of cryptography was not needing to keep your algorithm a secret, but publishing the algorithm and letting the academics go crazy and write Ph.D. theses and then having a key which you keep secret. And so the algorithm is keyed.

Leo: And in contrast, I remember reading that one of the things SolarWinds executives often pooh-poohed is the security of open source software, saying we're closed source, so you can trust us. Oh, well.

Steve: That's right.

Leo: Oh, well. Great show, Steve. I know the bad guys took a break last week. Don't worry. They're coming back. No fear. Oh, boy, are they coming back.

Steve: Yeah, they want to keep us busy here on Security Now!.

Leo: Yeah, yeah. They're providing fodder. We do this show every Tuesday, right after MacBreak Weekly, just about 1:30 Pacific, 4:30 Eastern, 21:30 UTC. If you like to watch us do it live - which has its advantages and disadvantages. You can see the before and after show chatter, but on the other hand all the swear words are still in and - no, no. We don't really edit the show very much.

Steve: All of Steve's hiccups are still in.

Leo: All the hiccups are still in it. You can watch the livestream at TWiT.tv/live. It's behind the scenes, the making of, that kind of thing. If you're watching live, you should chat with us live at irc.twit.tv. On-demand versions of the show are available at Steve's site, GRC.com. He's got 16Kb audio, 64Kb audio, and transcriptions, which is very nice, plus the show notes: GRC.com. While you're there, pick up a copy of SpinRite. If you buy 6.0 now, you'll get an upgrade, free upgrade to 6.1, plus you get to participate in the development of 6.1, which as you can see is quite active right now.

Steve: Yes, baby. That's beginning now.

Leo: Yeah. We have 64Kb audio and video at our site, TWiT.tv/sn. There's a YouTube channel devoted to Security Now!. And of course you can subscribe in your favorite podcast client, and that way you'll get it the minute it's available of a Tuesday afternoon. Our TWiT survey is on. We do this once a year at the beginning of the year. As you know, we don't want to, nor do we have the ability really to gather information about who listens to this show. We make it voluntary. And it's helpful for us in selling the show and telling potential advertisers who's listening and so forth. If you want to participate, the survey is online now at TWiT.tv/survey21. It's our 2021 survey. Shouldn't take more than 15 minutes.

I know a lot of you feel like "I've got to complete this. I've got to get every answer." You don't have to answer every question. Answer it as best you can. We always will get the people saying, "Well, I use four different operating systems. You only mentioned three." Don't worry about it. Do the best, you know, give us the best information you can within the limits of the survey. It is honestly very helpful. When we can usually get 10, 20, 30,000 people to take it, I think it's statistically valid and makes a big difference in selling. So thank you. That's what keeps us on the air, of course: TWiT.tv/survey21.

Steve, I've now embarked on Volume 2 of the Salvation Trilogy.

Steve: Yay.

Leo: Really good. You were absolutely right. And, boy, on Sunday, I was berated by our TWiT panel for not having watched "The Expanse" yet. So I started that last night. You're right, it's quite good.

Steve: It is. You need to, you know, it's a complex plot. It's a little political with the Belters and the Martians and the Earthers. But, oh, and boy, wait till you get to - like some of those battle scenes, they did some of the best space battle stuff I've ever seen made.

Leo: I'm watching it and thinking, how are they doing the weightlessness stuff? Are they underwater? What are they doing?

Steve: It's really good.

Leo: It's very realistic. Maybe they fly the Vomit Comet. I don't know. But it really looks - it looks good. And the CGI is excellent, and it's a great script. Of course, I have the novels, which you long ago recommended.

Steve: Yes.

Leo: And I'm torn whether I should read the novels before I watch the show. But I guess I'll watch the show and then read the novels.

Steve: I think you should watch the show. I think that what the novels do is serve to anchor it. It's so often the case that the best movies come from good writing of an underlying novel.

Leo: Yeah, yeah. "The Expanse," that's on Amazon Prime. And it's beautiful. I decided to watch it on the 4K TV; and, man, it looks good.

Steve: I forgot, I didn't ask you whether you had a chance to see "In the Shadow of the Moon."

Leo: Not yet, not yet, not yet.

Steve: Okay.

Leo: I wrote it down. It's on the list. Can't wait.

Steve: No problem.

Leo: Yeah. All right, Steve. Have a great week, and stay safe. We'll see you next week on Security Now!.

Steve: Bye.