GIT Me Some PHP

Description: This week we begin by checking in on the patching progress, or lack therefore, of the ProxyLogon Exchange Server mess. We examine a new Spectre vulnerability in Linux, a handful of high-severity flaws affecting OpenSSL, still more problems surfacing with SolarWinds code, an intriguing new offering from our friends at Cloudflare, and the encouraging recognition of the need for increasing vigilance of the security of increasingly prevalent networked APIs. I'll check in about my work on SpinRite. Then we're going to take a look at the often breathlessly reported hack of the PHP project's private Git server, and why I think that all the tech press got it all wrong.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-812.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-812-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. There is a lot to talk about. Steve's got a unique take on the PHP Git repository hack. He thinks it's actually a good thing. We'll find out more about that. A virtual browser solution from our friends at Cloudflare that looks pretty useful. And a little plug for our friend Rasmus Vind and his Warcraft site. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 812, recorded Tuesday, March 30th, 2021: GIT Me Some PHP.

It's time for Security Now!, the show where we cover the latest news about the world of security. And, boy, is there some news this week. But I guess you could say that about every week. Steve Gibson's here from the GRC company. When we first started doing this 812 episodes ago we thought, how will we ever fill half an hour? And now we fill four half hours, and there's still more to do.

Steve Gibson: Oh, my goodness, yes.

Leo: Hi, Steve. Did you have a good week?

Steve: Had a good week, yes. Got some work done, lots of work actually, on SpinRite.

Leo: Yay.

Steve: I'll mention it briefly toward the end.

Leo: People are happy to hear that.

Steve: This 812th episode I titled GIT Me Some PHP.

Leo: And I know why.

Steve: Yeah. And my reading is that all of the tech press got it wrong.

Leo: Oh. Good, that's why we count on you. Good.

Steve: Yeah. As I was reading all the huffing and puffing, I thought, you know, it's not clear to me that this was anything more than what will end up being a big benefit. But we'll get to that. We're going to begin by checking in on the patching progress, or lack of course thereof at this point, of the ProxyLogon Exchange Server mess. We examine a new Spectre vulnerability to hit Linux, believe it or not. So Spectre's not over. We have also a handful of high-severity flaws affecting OpenSSL, probably one of the other things you were thinking of, Leo.

Still more problems surfacing with SolarWinds code. An intriguing new offer from our friends at Cloudflare, who just, you know, innovation seems to be their middle name. They keep adding stuff. They're not sitting on their laurels, that's for sure. And we've got some encouraging recognition of the need for increasing vigilance of the security of increasingly prevalent networked APIs. Then, as I said, I want to briefly just touch on my ongoing work with SpinRite. And then we're going to take a look at what was a breathlessly reported hack of the PHP Project's, like, main central Git repository server, and why I think all the tech press got it very wrong.

Leo: Interesting.

Steve: And we do, of course, have kind of a fun Picture of the Week. So I think another great podcast for our listeners. And, yeah, no sign of security problems letting up.

Leo: We should do a traffic report. And it's all jammed up along the information superhighway.

Steve: On the 405, yeah.

Leo: Okay, Steve. I'm ready with the Picture of the Week. This one speaks to me. It does.

Steve: I had a feeling it would.

Leo: Yeah, it really does.

Steve: So what we have is two service windows. One, there's a big signage over the one on the left says "More Gear." Get your more gear here. And then the other window next to it says "Learning to Use Existing Gear."

Leo: Guess which one has the longer line?

Steve: Yeah, the guy behind the Learning to Use Existing Gear, well, he's not asleep because he's reading a book with his head propped up on his hand. There's nobody who has any interest in learning to use their existing gear, apparently. There's nobody at his window. The entire line, and it goes right off the screen, is just I need some more gear. Give me some more gear.

Leo: Don't learn to use what you've got. Get something new. It'll be more easier. It's not signed, but it really looks like a Rich Tennant cartoon. You know, he did "The 5th Wave," it was in the "For Dummies" books, it was in magazines like - I can't remember, it was PC - no, it was Computerworld, that's what it was. Rich Tennant. Really looks like might be one of his.

Steve: You know, and I have to say I often harken back to when I was five because I knew what every button on every piece of equipment that my family owned did. I mean, I just - that was my thing. It made sense to me.

Leo: Yeah, that's how you got to be a geek.

Steve: I was going to know what it is. But I look at the remote control on the whatever it is I've got, and there's buttons there. You've got your top menu, your side menu, your backwards menu, the this and the that, I mean, it's just like, what is all this? Do I really - all I want to do is go to the next episode of "Fringe."

Leo: It's a lot harder than it used to be. I agree with you.

Steve: Oh, Leo.

Leo: I don't know if that's us. I don't think it is.

Steve: I don't know.

Leo: I just think everything has proliferated to such a degree. And user interface design seems not to have improved enough to accommodate all the new features.

Steve: Yeah. I mean, even, you know, one of the most elegant things about the iPhone was that there was four buttons. You clicked a picture, and you got a phone. Or you clicked when you got notes. You know, they did have a little wood grain problem there back in the beginning. But still.

Leo: That was to make you feel at home.

Steve: There just was, like - now it's like, oh, you've got to push the screen harder and hop on one foot and then do a little twist with your finger in order - and then you get some secret menu that you never knew was there. And oftentimes, I mean, and I'm certainly no guru, but I'll do something, like I'll swipe along the bottom and go back to an app because I saw that on one of your shows once, Leo. And Lorrie goes, how did you do that? What's that? What? What?

Leo: Yeah, there used to be a home button. They took it out. How do I get out of here?

Steve: The other day I was helping her with something in Chrome, and she was at some link somewhere, and I grabbed to the left of the URL and dragged it off and dropped it on the desktop. And she's like, "What? That's a thing?" It's like, "Yes, honey. Now you have an icon that will get you back there in time." She says, "Oh," she says, "I just need to watch you work more often." But again, so yeah, I have like four things...

Leo: It's just as baffling for us, though. That's the thing.

Steve: Yes. I have four things that she hasn't figured out. But I look at this stuff, and I think, what is hidden behind all these user interfaces?

Leo: I feel bad, I mean, if you and I have trouble, what's a normal person to do, really? And I think that that's why this cartoon is so great, because I think really normal people just - they don't try. I don't know how to get better using this, so I'm just going to get another one, which I'll be just as lost in. Oh, gosh. I really feel bad for people.

Steve: Okay. So our weekly ProxyLogon update. I looked for any update from Microsoft, from RiskIQ, which is the people that they keep citing, or any other source to get some sense for how the patching was going. I did discover that RiskIQ is now estimating that "only" - and I put that in quotes. "Only" needs to be in quotes because the only sense in which it's "only" is in comparison to the original several hundred thousand vulnerable and unpatched Exchange servers that we started out with. So today we're down to "only" 29,966 instances of Exchange Server still vulnerable and thus still wide open to attack. But that's down from the last number we reported, which was 92,072, back around March 10th. And that nearly 30,000 number appears to be holding.

And of course unfortunately our experience suggests that, especially at this point, right, like anyone who was going to get the news, got the news in the last three weeks since this happened. So four weeks, actually, yeah, one, two, yeah, four weeks now exactly since March 2nd. So we know that any further improvement will be incremental, slow, attritional, and perhaps the result of Microsoft's slipping that useful ProxyLogon remediation into their Windows Defender solution, since it's able to filter the primary exploit vector from the IIS web server before that attack reaches the server's tender underbelly.

It's not clear whether RiskIQ is actually testing for the vulnerability, which I don't suspect, or obtaining Exchange Server's version information from some logon hello

handshake, which is probably what they're doing. So if it's the case, then RiskIQ's number would be high because it would not be crediting all of those instances of Exchange Server which had not been actually updated, but which did have the Windows Defender remediation slid in and is protecting it sort of without getting any credit for doing so. So it's probably the case that the number's coming down. But tens of thousands. And as we know, it's like there's a battle for who can take control of these servers and what mischief they can get themselves up to.

Leo: Well, and remediation doesn't kick them out. It just prevents new ingress. But if somebody's already in there, they're in there; right?

Steve: Yeah. Now, what they did say, without any specifics, is that it will also go and try to find the things that it knows of that they may have done.

Leo: Oh, good. Okay. So you could see traces of them, crumbs left behind.

Steve: Exactly. So it may have been able to remove them, which would also be a good thing.

Leo: Yes.

Steve: And I need to turn my - pardon me. I'm trying to get our balances correct now, having turned up my microphone, now I'm blasting myself.

Leo: I should explain we've used Skype since Episode 1, but we started using Zoom because we're concerned, well, first of all, you never liked, from the day Microsoft acquired them, the things they did to change how Skype worked. It used to be peer-to-peer. Now all of a sudden it's going through Microsoft servers, blah blah blah.

Steve: Yeah.

Leo: So after looking for many alternative solutions, Alex Lindsay suggested moving to Zoom. He's doing a lot of streaming, obviously, and he knows more probably than anybody about which works best. And we're very happy with Zoom. But it's a whole new set of interfaces and a whole new set of buttons that have to be tweaked.

Steve: Now it's all wonderful.

Leo: Yes.

Steve: You sound great.

Leo: Good.

Steve: And I'm just a sultry whisper in my own ears.

Leo: A whisper in your own ears. Good.

Steve: Yeah, and "sultry" is not a word that's ever been used to describe myself.

Leo: Actually, the chat room said you have a rich, velvety sound now. So, hmm.

Steve: Ah, well. Just be chatting. So Spectre is remaining with us. It's clear. Two weeks ago we noted that Google's security blog was titled "A Spectre Proof of Concept for a Spectre-Proof Web." And they demonstrated two weeks ago and shared their creation of a working Spectre exploit in JavaScript that's able to obtain data from the browser's internal memory. And yesterday researchers with Symantec's Threat Hunter Team disclosed two ways in which Spectre's processor mitigations could be circumvented in Linux-based OSes to launch successful speculative attacks to obtain sensitive information, like crypto keys, from the system's kernel memory.

And what's interesting about these attacks is that on the one hand, in the very beginning, the first time there was any notion of, like, oh, we could leak a bit every minute, people were like, who cares? A bit a minute? But crypto keys are notoriously dense; right? You need to get all the bits right, or you don't have anything. So that's, you know, every single bit is crucial. But they're not very long overall. And you really want to hide their bits well. Well, the one thing that this kind of low bandwidth leakage does is it does get things that you thought were well hidden exposed.

So the Symantec group found two related but different ways to pull off something like crypto key in the kernel memory leakage. Two CVEs have been assigned. And interestingly, they're 2020: 2020-27170 and 27171. Because these should be fixed, but they do not spell the end of the world or of Linux, they carry relatively mild CVSS scores of 5.5. And they impact all Linux kernels prior to v5.11.8. So the trouble was first identified last year, thus the 2020 CVE years. And the Linux teams were notified. Patches for Ubuntu, Debian, and Red Hat were just published on March 17th, and then they were released for deployment the Saturday before last, on March 20th.

So as I said, there are two. The first one is able to reveal the contents from the entire memory of an affected computer; and the second one, 171, can reveal the contents from the 4GB range of kernel memory. So remember that, because Spectre and Meltdown are chip-level vulnerabilities, operating system patches can only be mitigations, which are designed to make it hopefully impossible for an attacker to exploit the vulnerabilities.

My point is that the operating system has no ability to address the underlying issue which exists in the processor beneath it, that it can't get there. It is able to load microcode at boot time. And Intel has updated the microcode, and the Linuxes are carrying that, and they are indeed doing as much as Intel has been able to do. But it's these mitigations for Spectre which were also incorporated into Linux, which the Symantec group found a way to get around, to essentially bypass the mitigations. So by using these mitigation bypasses on any unpatched Linux before - and I wrote before 5.11.7, but it must be before or including 5.11.7, since the update brought us to 5.11.8 - so then malicious code can read memory that its process, that is, the process the malicious code is running in, should have no permission to inspect.

And what's more, the attacks could also be launched remotely through malicious websites running exploit JavaScript. So when you hear that, that's the concern; right?

Because we've often talked about how, yeah, Spectre and Meltdown, they're not good. We don't want to have our processor leaking between processes. But still, it's more of a theoretical than a practical issue, we have been led to believe for the last couple years. And the point has been that, if you've got a process sharing your personal system, that using Spectre or Meltdown, you've already got bigger problems than its ability to perform some difficult-to-execute memory leakage. But if it's possible for JavaScript running in your browser to do this when it's on a Linux machine, then that's something to worry about.

So the Symantec team worked out a way to take advantage of the kernel's support for the Extended Berkeley Packet Filters known as EBPF, to extract the contents of the kernel memory. The Berkeley Packet Filter, BPF, has been around forever. It started out to be a general purpose, lightweight virtual machine that was used to inspect the contents of network packets, the idea being that, if you wanted to make a fancy packet inspector, you need a little bit of code to do that, to perform some pattern matching. If this is that, then check if this is that and, you know. It's a little more difficult than you could do with a set of fixed rules.

So they implemented a simple virtual machine, the Berkeley Packet Filter, in order to perform those sorts of simple things and to make them very fast because you don't want something slowing down your packets. So anyone who's ever had occasion to use Linux's TCP dump facility has likely encountered the BPF system. Since then, the Extended BPF variant has become a universal in-kernel virtual machine which has hooks throughout the kernel in order to do what it needs to get done.

So Symantec explained, they said: "Unprivileged BPF programs running on affected systems could bypass the Spectre mitigations and execute speculatively out-of-bounds loads with no restrictions." And that's the no-no which Linux was trying to prevent. They said: "This could then be abused to reveal the contents of the memory through Spectre-created side-channels." And specifically it's in kernel/bpf/verifier.c. They said: "The kernel [that file] was found to perform undesirable out-of-bounds speculation on pointer math, thus defeating fixes for Spectre and opening the door for side-channel attacks."

So the point is that Linux has been hardened against Spectre. But there was a little piece that didn't get hardened, that Symantec realized, oops, you could still do this. So in a real-world scenario until recently, as we know, Spectre has been a bit light on unprivileged users' ability to leverage these weaknesses to gain access to secrets. But this allows a way for that to happen.

Symantec explained that: "The bugs could also potentially be exploited if a malicious actor was able to gain access to an exploitable machine through some previous step, downloading malware onto the machine to achieve remote access. This would allow them to further exploit these vulnerabilities to gain access, for example, to all user profiles on the machine."

And again, patches are out. So individuals should have no problem updating themselves, which you'll want to have done since the 20th. But, you know, here we were just talking about the difficulty of getting the world's systems updated, the world's Windows systems updated. Just imagine how many Linux systems have not been updated in the past two weeks. Many haven't been updated in years, and won't be, never will be. So Symantec tells us that these unpatched gremlins that they have found can be exploited remotely, unfortunately.

Leo: Is it a firmware update? Or is it an operating system update?

Steve: It's an OS. It's an OS update.

Leo: Okay.

Steve: So the problem actually existed in some lack of remediation against the remaining ways that even after the firmware was updated, that it was still possible to do this. So it's in that C file.

Leo: Okay. All right. Well, yeah, I'll do the update. The problem with Linux, well, you know, as you point out, that a lot of these boxes are designed just to run forever and not be updated. I have a server in the other studio that, you know, it's a server. I don't want to update it more than necessary, and it doesn't update automatically. A lot of people who have desktop Linuxes, in fact I've seen people complain about this, are obsessively updating, like every day they update. But that still doesn't guarantee you're going to get the update because the way Linux distributions work, the updates happen upstream, and then they have to be incorporated into the distribution so that your distribution, whatever it is, will see it. Some distributions are slower about that than others. It's not a uniform process. So, interesting.

Steve: Well, and how many appliances, how many turnkey boxes of one sort or another.

Leo: Yeah. Oh, gosh, yes.

Steve: I mean, all of the routers, they're all Linux-based.

Leo: Right, right. As are, I mean, anything Android-based is Linux-based. So there's a lot of Linux out there, yeah.

Steve: Yeah.

Leo: By the way, I don't know if you noticed, but I'm glad you didn't come to me earlier because I put in new rubber bands in my - Burke said, oh, I got some. Look at that.

Steve: Oh, perfect.

Leo: It's bouncing right in there. Look at that.

Steve: Oh, nice.

Leo: It's only been that way for, like, four years. Yours too, probably. On with the show.

Steve: I've not touched it in 15 years of the podcast.

Leo: Yeah, exactly. They go in about three, so that'll give you some idea. On we go.

Steve: So the OpenSSL project has fixed several high-severity flaws. Alarm bells were also ringing over at the OpenSSL project as a result of a server crash Denial of Service and a certificate verification bypass. So as we know, for many years, OpenSSL contained the main repository of open source crypto magic, so the OpenSSL library was incorporated everywhere that secure communications and certificate management was needed. Again, don't reinvent the wheel. Security, especially security code is hard to get right, so just drop in the library. And the library that was dropped in was OpenSSL.

Now, these days crypto's gone much more mainstream, and OpenSSL now has many viable newer and quite a bit sleeker competitors. We've talked about Bouncy Castle, Cryptlib, GnuTLS, Libgcrypt we were just talking about recently, Libsodium, NaCl, NSS, I mean, there are many alternatives now. And even Amazon created a super svelte TLS implementation for their own AWS stuff.

Leo: You have one you prefer. I think you used NaCl for SQRL; right? Or was it Libsodium?

Steve: Libsodium, actually. But the two are almost - Libsodium and NaCl are...

Leo: NaCl is salt; right?

Steve: Right, right. So inertia being what it is, OpenSSL remains dominant. So it's under most of the rocks that you will turn over. It is big. It's bloated. It's creaky. But it remains the reference standard against which the performance of everything else is compared. If you're creating a clone function, you see what OpenSSL does, and then you make sure that yours does the same thing. So although it has by any measure through the years been quite robust and secure, its popularity means that, when something goes wrong, it's generally a pretty big deal.

The biggest previous mess brought to us by OpenSSL was a worrisome little flaw that became known as Heartbleed. Ouch. And any of our listeners from seven years ago will appreciate what a ruckus Heartbleed created back in 2014. What the two recent discoveries lack, probably, is marketing. Somehow naming this CVE-2021-3449 just isn't nearly as catchy as Heartbleed. And there's no wonderful dripping blood logo. But it is still quite worrisome. And I think we've probably not heard the end of it.

Last Thursday morning cryptographic engineer Filippo Valsorda tweeted: "CVE-2021-3449 looks like it could have been found easily if anyone figured out how to fuzz renegotiation." But, he said: "Renegotiation is sadness." He says: "Anyway, sounds like you can crash most OpenSSL servers on the Internet today." And that is true. Bottom line, this lets you crash most OpenSSL servers, which is to say most Linux-based and Unix-based servers.

Okay. So that brings us to last Thursday's OpenSSL Security Advisory from the 25th of March. It had two pieces. The first was NULL pointer deref in signature_algorithms processing. And it's describing this first of the two problems, 3449. They rated its severity as high. And they said: "An OpenSSL TLS server may crash if sent a maliciously

crafted renegotiation ClientHello message from a client. If a TLS v1.2 renegotiation ClientHello omits the signature_algorithms extension where it was present in the initial ClientHello, but includes a signature_algorithms_cert extension, then a NULL pointer dereference will result, leading to a crash and a denial of service attack."

They said: "A server is only vulnerable if it has TLS v1.2 and renegotiation is enabled, which is the default configuration. OpenSSL TLS clients are not impacted by the issue, only servers." And they said: "All OpenSSL 1.1.1 versions are affected by this issue. Users of these versions should upgrade to OpenSSL 1.1.1k." They said: "This issue was reported to OpenSSL on the 17th of March 2021 by Nokia. The fix was developed by Peter Kaestle and Samuel Sapalski from Nokia."

Okay. So note that the advisory just told any malicious prankster how to down most of the Internet's servers that use OpenSSL to provide TLS v1.2 support, which is pretty much everything today.

Leo: Crikey.

Steve: So that's why I said I don't think that we have heard the last of it.

Leo: That's not good.

Steve: No. The good news is taking servers down hopefully is less gratifying today than it would have been 15 years ago. Today they want to crawl in there. They want to set up their cryptocurrency miners. They want to...

Leo: Yeah, it's just malicious malarkey versus actual valuable stuff.

Steve: Right. Still...

Leo: There's still malicious malarkey out there.

Steve: It is. And having servers down can be pesky. So if anyone notices that their servers are crashing suddenly for no obvious reason, well, you want to update your OpenSSL.

Leo: I'm logging in right now.

Steve: Okay. Now, that seemed a little tricky; right? The other OpenSSL problem that was also fixed last Thursday could best be described as a weirdo edge/corner case that you'd really need to try hard to create. But if you did, the result would be a true bypass of certificate verification in OpenSSL. And that would obviously be very bad since, if you cannot authenticate the identity of the party you're having a private conversation with, it doesn't really matter if it's a private conversation. It could be a man in the middle or anyone that you're actually talking to.

So as I was writing this, I thought, I'm tempted to share the advisory's description just so you'd have a clear example of exactly what a "weirdo edge/corner case" exactly sounds like. And then I thought, oh, what the hell. Here's how. The advisory describes the problem that they also fixed. This is CA (Certificate Authority) certificate check bypass with X509_V_FLAG_X509_STRICT. Also severity high. Again, this is an authentication bypass for the certificate chain in OpenSSL and anything you use it for. So not good.

So here it is. They write: "The X509_V_FLAG_X509_STRICT flag enables additional security checks of the certificates present in a certificate chain. It is not set by default. Starting from OpenSSL version 1.1.1h, a check to disallow certificates in the chain that have explicitly encoded elliptic curve parameters was added as an additional strict check. An error in the implementation of this check meant that the result of a previous check to confirm that certificates in the chain are valid CA certificates was overwritten. This effectively bypasses the check that non-CA certificates must not be able to issue other certificates." Whoops.

"If a purpose" - as in one of the declared purposes for the certificate. "If a purpose has been configured, then there is a subsequent opportunity for checks that the certificate is a valid CA. All of the named purpose values implemented in libcrypto perform this check. Therefore, where a purpose is set, the certificate chain will still be rejected, even when the strict flag has been used. A purpose is set by default in libssl client and server certificate verification routines, but it can be overridden or removed by an application.

"In order to be affected, an application must explicitly set the X509_V_FLAG_X509_STRICT verification flag and either not set a purpose for the certificate verification or, in the case of TLS client or server applications, override the default purpose. OpenSSL versions 1.1.1h and newer are affected by this issue. Users of these versions should upgrade to OpenSSL 1.1.1k," the one that just came out. And this issue was reported to OpenSSL on the 18th of March by Benjamin Kaduk from Akamai and was discovered by others at Akamai. The fix was developed by Tomas Mraz.

So we're not going to lose any sleep over that one. It must have been, you know, this is not something you would discover like in the wild or in the field or anywhere. It must have been that the guys at Akamai were perusing the OpenSSL source because, again, this was only introduced in .1h, and we're on "k." So they must have been looking at the source and spotted the logic flaw that way. It's never good, as I said, to have any way around authentication in a system whose entire purpose is authentication. So it'll be good to have this one resolved. But there are a bunch of servers out there.

It's unlikely that the situation exists that actually allows this to be triggered in the wild. And that "h" subversion came out last September. So the window of opportunity, like from September till now, "h" through "k," is just not that wide. It's nothing like the 11-plus years during which Exchange Server has had these flaws, thus all Exchange servers, even ones out of currency, are vulnerable to the Exchange Server problems. But still it's good to get these things patched. I'm sure that when we issue our command to check for libraries in Linux or Unix that need to be updated, OpenSSL will now pop up. And it's like, yep, let's get that code updated.

SolarWinds keeps finding new critical problems within its own code. Last Thursday was a busy day. SolarWinds released a new update to its Orion networking monitoring tool to fix four security vulnerabilities, including two that could be exploited by an authenticated attacker to achieve remote code execution. So that's better than unauthenticated, but perhaps not enough better. We've talked about JSON deserialization flaws, about how deserialization inherently requires interpretation, and how difficult it is to create perfectly robust interpreters. The programmers who write these serializers, and that's something

that turns a dense data structure into some sort of a series of bytes which you can then store, and then later you deserialize in order to restore the original data structure.

Invariably the guys who wrote the deserializers are the same ones who wrote the serializers, or at least the spec, the serialization spec. And the assumption is too great that the data that the deserializer will be receiving came from the serializer that the same guys wrote. So the point is you just make the assumption that valid data is what you're being asked to deserialize. And we have seen time and time again that that results in vulnerabilities which create buffer overruns which end up being critical, must-fix-now problems. And the Orion Web Console has one of those.

The second issue concerns a high-risk vulnerability that could also be leveraged by an adversary to achieve remote code execution in the Orion Job Scheduler. The release from SolarWinds notes: "In order to exploit this, an attacker first needs to know the credentials of an unprivileged local account on the Orion Server." So not privileged, but at least some credentials required. And both of these came from Trend Micro. There are also two others, a high-severity stored cross-site scripting vulnerability in the "add custom tab" within the customized view page and a reverse tabnabbing - we've talked about that in the past - and open redirect vulnerability in the custom menu options page, both of which require an Orion administrator account for successful exploitation.

So it does sound like the really bad egregious problems are - they are no longer finding those. So it brings a number of other improvements and fixes along the way. But, you know, as I'm thinking about SolarWinds and how bad a problem they've had, how many problems have been fixed, it sort of begs the question, I think, that certainly many people in government and industry must be asking themselves: Should SolarWinds now be abandoned for a hopefully more secure alternative? The key of course is whether an alternative would truly be more secure. It could be that with all the hot water that SolarWinds has recently been in, their code finally got the deep cleansing security scrutiny that it had always needed, so that now it's actually the better solution compared with the others that perhaps haven't had the scrutiny that SolarWinds' time in the spotlight has given it.

It's somewhat like the dilemma that an employer faces after discovering some errant action of an employee who sincerely apologizes after being called onto the carpet for it. Is it better to then sever the transgressor's employment over that mistake, or are they now a better employee for having learned a valuable lesson? Again, the age-old dilemma. In the case of SolarWinds, my feeling is that bad code somehow got in there in the first place, and it wasn't found.

So to keep me as a customer in the long term - and I'm not a customer of SolarWinds, but if I were - to keep me I would need to be convinced that not only were all of this handful of flaws patched up and fixed, and that's good, but that the flawed system that created them in the first place had also received what was apparently some much-needed attention and patching. So tough to decide whether you leave something that's been fixed because it was once broken, or think, well, now it's fixed, so the devil you know.

Cloudflare is continuing their move toward offering more and more security-related services. Last week they announced and debuted a web browser virtualization service as part of their Cloudflare for Teams offering. They call it "Zero Trust Browsing." I just really like the things that these guys are doing. Their description explains the motivation behind it. They said: "Cloudflare's Browser Isolation service makes web browsing safer and faster for your business, and it works with native browsers. Web browsers," they said, "are more complex and sophisticated than ever before." And, boy, is that a theme of the podcast. They said: "They're also one of your biggest attack surfaces." Again,

hello, yes. "Cloudflare Browser Isolation is a Zero Trust browsing service. It runs in the cloud away from your networks and endpoints, insulating devices from attacks."

They said: "Secure Web Gateway policies are too restrictive, or too relaxed. No secure web gateway can possibly block every threat on the Internet. In an attempt to limit risks, IT teams block too many websites, and employees feel overly restricted. Then there's malicious content, which is difficult to spot and costly to remediate. Innocuous webmail attachments, plugins, and software extensions can disguise harmful code. Once that code travels from a user's browser to their device, it can compromise sensitive data and infect other network devices." And they wrap up, saying: "IT teams have limited power to manage browser activity. Organizations often do not have full visibility into or control over the browsers their teams use, keeping them from meeting compliance standards and securing the users, devices, and data over their network."

So we might think of it like Remote Desktop for browsers. But the desktop is not being remoted. Only the browser's fetch and render engine is remote. The browser's network communications, all the stuff it fetches, all the interpretation it does, the scripting it runs, the rendering it does, all live at Cloudflare. And Cloudflare sends the rendered visual result, and only the visual result, to the user's browser. And apparently they're able to pull this off so that the lag is negligible, unnoticeable.

And I was thinking about this. You know, given how insanely complex today's web pages have become, reaching out to so many differing third-party servers to pull page sub-assets, it does make a certain sort of sense to outsource that entire process, that entire machine, to a capable and well-connected cloud provider like Cloudflare. Their DNS servers can have massive caches to minimize the need for lookups. And we know that, when you share a big cache, a big DNS cache, with a lot of people, the IPs you're looking for are already going to be in the cache. So that's a win. And in fact they can also have massive caching proxies for the Internet. Which means that everything can be network-local to that browser cloud engine. So you could theoretically render pages at lightning speed by dramatically reducing all lookups and network transit delays, blast the page together, then intelligently send the post-rendered page result to the user, thus completely offloading all of that work and protecting the user.

And of course the whole point of this is that anything that attacks the browser is then also remote, since there's not any system to attack at the remote end, just this browser, this virtual browser. And the only thing the user receives are post-digested page image results. And it's interesting also because, by the end of today's podcast, where we'll be talking about the hack of the PHP project's private Git server, we wind up looking at the growing trend toward outsourcing of services for which little local value can be added. If we cannot add any value to a service, why do it ourselves, especially if there's a downside.

And when you think about it, why are we all pulling all of these disparate web browser assets redundantly from all over the Internet to each of our own individual web browsers? It really does make a sort of sense to imagine having a "browser service" that does all of that non-value-addable redundant work for us, then sends us only a safe, attack-free, already digested final result. It's going to be interesting to see how this evolves. If anyone's curious to learn more, I have a link to the Cloudflare page describing their new browser isolation feature in today's show notes. It's Cloudflare.com/teams/browser-isolation. Really sort of an interesting idea, I think.

And I just wanted to sort of plant a flag on the issue of API security, a report that was recently out from, not surprisingly, a company that is selling API security. So there's that. But it had some interesting stats, and it is a thing. So the original concept of an API didn't need any security. There was no such thing as API security. It was entirely local. Operating systems offered their underlying services through calls to operating system

functions, like asking the OS to launch a process to allocate some memory, to open a file and read its contents. And because there were operating system applications, and programmers used these service calls, over time they became known as application programming interfaces, APIs. And the operating system was then sent to be the publisher of these interfaces.

So generically what evolved was the idea of carefully and clearly defining a set of function calls that one entity would publish, meaning to offer, which would then be consumed or used by one or more API users or consumers. The big change then happened with networking, the introduction of networking. It occurred to developers of increasingly sprawling systems and solutions that whereas web browsers had traditionally been using HTTP queries and responses to obtain things to show on the page, there was no reason why the parameters used by traditional local operating systems and other application APIs, which were typically binary parameters, could not be turned into well-formed text and sent over the wire in exactly the same fashion as HTTP web traffic. So network APIs were born.

The problem is that insufficient attention has been given to the security of publicly exposed APIs. And consequently, attacks against APIs are another area of growing malicious interest. So this is forcing enterprises to start taking the security aspects of API adoption more seriously. So the good news is the need for security is on people's radar. And according to this report, 91% of the IT professionals they surveyed claim that API security should be considered a priority over the next couple years, especially since more than 70% of enterprises are estimated to be using more than 50 different sets of APIs.

The main aspects of API security which respondents considered to be a priority is access control, which was cited by 63% of those, and I'm surprised that number's so low, I mean, like access control is everything; regular testing, 53%; and anomaly detection and prevention, 43%. Again, I'm not sure why all that's not 100, but okay. So maybe someday. In total, eight out of 10 IT admins want more control over their organization's APIs, like sophisticated API-aware firewalling. Yet tools for that are currently kind of lacking. And then a couple other stats jumped out at me: 19% of enterprises test their APIs for signs of abuse. Okay. Meaning that 81% don't? Four out of five organizations enable their partners or users to access data using external APIs; right? That's not surprising, 80%. That's often what these APIs are doing. They're information-sharing APIs.

The current focus of network API strategies are centered around application performance and development and integration. And, finally, 64% of survey respondents said their current solution is to not provide robust API protection. So anyway, there's no takeaway for us at this point. But I just wanted to put it on everyone's radar, as I said. We're seeing an ever-increasing amount of automation. IoT is all about, I mean, like IoT is networked APIs. When I've got an IoT thermostat, and I've got a humidity reader, and I've got a few of those AC plugs on timers, that's all network API. And so it's going to explode with the continuing explosion in IoT. So I have a feeling that we'll be talking about exploits explicitly against networked APIs in the future.

So SpinRite. The work on 6.1 is moving nicely forward. And although in one sense - and now I'm gaining like experience with this conversion; right? In one sense it's the same SpinRite, but with direct maximum performance hardware support for IDE and SATA drives through ATA and AHCI interfaces. Doesn't sound like a big deal. The implication, though, is that sector addressing is expanded from 32 to 64 bits. Since it was the 32-bit sector addressing that clamped all previous SpinRite at 4.3 billion sectors. Right? 4.3 billion. We're running across that number all the time. That's the number of 32-bit IP addresses on the Internet.

Well, that's also the number of sectors you can address with 32 bits, 4.3 billion. And back in '04, when I finished with SpinRite 6, that was all we were ever going to need; right? Uh-huh. Well, that's only 2.2TB. So for SpinRite to be able to run on today's drives, meaning all of today's drives, I am needing to support 64-bit sector addressing. And since sector addressing is SpinRite, I am needing to update everything.

But I'm very happy with the way it's coming along. Before I began, I worried that it wasn't going to be any different, and that SpinRite 6.0's users, upon getting 6.1, might think, what did I wait all this time for? But in the process of moving through the code, I am making many improvements. So SpinRite 6.1's users will definitely notice a different-looking SpinRite. Many places where, I mean, I've had to rework things because the underlying plumbing had to be reworked. And while I'm at it, I'm making it better. So anyway, we're getting there. And I'm very pleased with the way it's coming.

Leo: Yay.

Steve: And Leo, I am very pleased with the way this podcast is coming.

Leo: It's rolling right along now that the microphone solutions have all been applied and all that stuff. Hey, I was going to mention, we interviewed on Triangulation about a year, two years ago a guy named Scott Petry. In fact, I remember Scott because he was a former Newton engineer, and I have all this Newton stuff that he sent along. But he had a company called Authentic8, with the number 8, that did virtual browsing. It was the same idea as Cloudflare's doing, where you would use their browser in the cloud and render it locally. So I wonder if Cloudflare acquired them, or maybe they just didn't - it wasn't unique enough. They call it Silo. I remember playing with it at the time and thinking, that's a pretty cool idea.

Steve: Well, and of course one of the offshoots of Chromium, right, is that rendering is now open source. So, you know...

Leo: Right, right, right. You could render, yeah. I think eventually a lot of what we do in computing, including running Windows, is going to be done that way, just run on servers.

Steve: Microsoft is talking about it.

Leo: Yeah, they already have it, virtual desktop.

Steve: I mean, they're like, yeah.

Leo: I think it's where they're headed.

Steve: Don't you worry about those pesky bugs. Just, you know...

Leo: Well, that's a good candidate for it. Let them update it. Let them deal with the flaws and all that stuff. It's on their servers; right?

Steve: Yeah. And if you can't run Windows for an afternoon because of a widespread outage, that's really okay; you know?

Leo: Take the day off.

Steve: We should walk more.

Leo: Well, there's already - you probably aren't too aware of it, but there's already several gaming services that do this. They have GPUs in the cloud. Google has one called Stadia. Microsoft has xCloud. Gaikai was bought by PlayStation Sony, and they do that. There's GeForce now. And all of them they rent - they have powerful machines in the cloud. And you can use an iPhone or an Android phone to get Triple A gaming because all the work's done remotely. So we've...

Steve: Ah, nice.

Leo: Yeah. We've seen this now, and I think this is kind of going to be the, like, big trend in computing. You're right. When Azure goes down, not so good. And not that that ever happens. Okay, Steve. I'm ready to learn all about PHP.

Steve: So I read all the coverage of this in the tech press. And I've looked at the source materials. And no one appears to understand that this had to primarily be a joke hack.

Leo: Oh. Not malicious, but a joke.

Steve: See if you don't think so by the time I give you my perspective. I think it was perpetrated by someone who arranged to compromise either the PHP Project's private Git server, as they believe, or the account of someone. Perhaps I'm missing something. But everyone appears to be taking this like a super serious attempt to actually sneak a backdoor into PHP. I don't think that's...

Leo: They committed updates that had backdoors in them, basically; right? No?

Steve: Yeah. Okay. So the code is a backdoor.

Leo: Okay.

Steve: Sort of. I'll explain that in a minute. But to the degree that it's a backdoor, it's not some sneaky, stealthy backdoor hiding in the shadows.

Leo: Yeah, in fact, PHP says, well, we noticed and fixed it within minutes; right?

Steve: Well, it's a backdoor embellished with big neon signs reading, "Hey, check out this wide-open backdoor I just created here."

Leo: Oh, oh, okay.

Steve: So, yeah, it's a backdoor, but it's screaming to be found.

Leo: Yeah, yeah.

Steve: Okay. So here's the code that was submitted. I've got the code in the show notes, and I know our listeners can't hear it, but I'll explain what the code does. So as we know, every browser query to a server identifies the browser, and typically a collection of its add-ons which may have been added to the browser, by sending a user-agent header, U-S-E-R hyphen A-G-E-N-T colon space and then the value of the header. The PHP code that was inserted into the repository, which you've got onscreen now in the show notes, it extracts the value of the HTTP user-agent header from the http_globals array that was built by PHP to describe the query. It holds that string, or it holds that value, the value of that header in a string named "enc" which it had declared up at the top.

It then checks to see whether the first eight characters of the user-agent header value are zerodium, Z-E-R-O-D-I-U-M. If it finds that the user-agent header does indeed begin with the eight characters zerodium, it then feeds the rest of the string, skipping those first eight characters, as one would, into PHP's insanely dangerous eval function, which interpretively executes whatever PHP code is passed to it, which is whatever is contained in the balance of the string. And driving the joke home, as if the presence of the trigger string test for zerodium were not glaring enough, our hacker then tosses in a quoted string reading in all caps: REMOVETHIS colon space, and then it says "sold to zerodium, mid-2017." It's like, what?

Leo: It's an old exploit. Is that what he's trying to say?

Steve: Well, yeah, exactly. This person who put it in two days ago was trying to say somehow you've missed this for the last four years. Okay. The official PHP documentation for the eval function reads: "Caution." Caution, bold larger type. Then it says: "The eval language construct is [in italics] very dangerous..."

Leo: Flawed.

Steve: Exactly.

Leo: That's why we put it in.

Steve: Oh, exactly, "...because it allows execution of arbitrary PHP code." Then again, italics, all italics this sentence, "Its use thus is discouraged." But of course, but it's there; right? And they say: "If you have carefully verified that there is no other option than to use this construct, pay special attention," and now we're going to switch into italics again, "not to pass any user-provided data," now back to non-italics, "into it without properly validating it beforehand."

Okay. So of course feeding user-provided data into the eval function is precisely what this little glaringly obvious snippet of code does. But it's that it's so glaringly obvious, deliberately calling attention to itself with the all caps "REMOVETHIS" as in, what? Remove this before use? Or don't leave any of this in here since it's a hack that was sold to Zerodium many years ago? It makes no sense.

Leo: It doesn't execute. And by the way, evaluating it doesn't make sense, either. It doesn't do anything; right?

Steve: Well, okay. So first of all, Zerodium's CEO was not impressed by this. He tweeted that the culprit was a "troll." That's his word.

Leo: Yeah, yeah. That's accurate, yeah.

Steve: Commenting that - this is the guy, the CEO: "Likely, the researcher(s) who found this bug/exploit tried to sell it to many entities, but none wanted to buy this crap, so they burned it for fun." Okay, now, wait. What? Maybe he also misunderstood this. It was a commit to the PHP Git server two days ago. It's not like it's been hiding in PHP since 1950 and no one noticed it until now. I mean, that entire block was added, not, like, a few characters changed to make this happen. And also interestingly, I thought this was interesting, the name of the actual header being checked is HTTP_USER_AGENTT. It's got two T's on the end.

I checked in with Rasmus, Rasmus Vind, who is, as we know, my go-to guy for all things PHP, to verify that PHP would, in fact, populate the http_globals array with any and all client headers it found in the query. He wrote some code to demonstrate that it does. So we'd have to presume that using the deliberately misspelled twice, it's not just misspelled once because he's using it, and then he's also taking the size of it elsewhere, that using HTTP_USER_AGENTT with the extra "T" was the hacker's way of hiding the use of what is actually a custom header in a lookalike header that might go unnoticed maybe. I mean, I saw it right away. But on the other hand, I program in assembler so I look for details. But maybe you would miss it in a cursory scan.

It might also be that commandeering the actual HTTP_USER_AGENT header for this purpose, that is, Zerodium and then some code, could have unforeseen side effects like causing the query to be blocked elsewhere. And finally, in an exercise, I don't know, dry wit or maybe a twisted sense of humor, the hacker gave their commit the title "Typo Fixed" and in the detail just says "Fixes minor typo." But it's a block of code. I mean, anyone who looks at it sees a block of code that's been added.

So on the serious side, what we definitely had here was a true, completely unauthorized incursion into the PHP private Git server. Had it gone unnoticed, if a tiny tweak had been dropped in, for example, the damage throughout the industry could have been substantial. But it was designed to be seen. Like, first of all, "Typo Fixed," and it's a block of new code? So again, you can't possibly. And then this Zerodium with the all caps REMOVETHIS. Your eye goes immediately to that.

So yesterday Nikita Popov, a well-known software developer at JetBrains and an active open source contributor - he works with PHP, the LLVM project and Rust efforts - he posted under the subject "Changes to Git commit workflow." And he wrote: "Hi, everyone. Yesterday 2021-03-28" - so two days ago for us. He said: "Two malicious commits" - only because it was like that block was created in two pieces - "were pushed to the php-src repo from the names of Rasmus Lerdorf, the creator of PHP..."

Leo: The creator of PHP.

Steve: "...and myself."

Leo: Oh, wow.

Steve: He said: "We don't yet know how exactly this happened."

Leo: It sounds like the credentials were compromised.

Steve: I think it's a credential hack. But he says, for whatever reason, he says no. He says: "But everything points towards a compromise of the git.php.net server," he says, "(rather than a compromise of an individual git account)."

Leo: Oh. That's much worse, actually.

Steve: Yeah, exactly. He said: "While investigation is still underway" - and Leo, that is the point, what you just said. "While investigation is still underway, we have decided that maintaining our own git infrastructure is an unnecessary security risk."

Leo: Yes. No one else does, yeah, yeah.

Steve: "And that we will discontinue the git.php.net server. Instead, the repositories on GitHub, which were previously only mirrors, will become canonical. This means..."

Leo: Good. That's sensible.

Steve: Yes, "...that changes should now be pushed directly to GitHub rather than to git.php.net. While previously write access to repositories was handled through our homegrown karma system, you will now need to be part of the PHP organization on GitHub. If you are not part of the organization yet, or don't have access to a repository you should have access to, contact me at" - and he has his email address - "with your php.net and GitHub account names, as well as the permissions you're currently missing. Membership in the organization requires two-factor authentication to be enabled. This change also means that it is now possible to merge pull requests directly from the GitHub web interface. We're reviewing the repositories for any corruption beyond the two referenced commits. Please contact security@php.net if you notice anything. Regards, Nikita."

So this is all good. The PHP guys are taking the opportunity of this hack to move their work from their private server, where they have been responsible for much more than just the code it contains. They're moving to GitHub, where they only need to be responsible for the code it contains, and the GitHub folks get to worry about the security of all the rest of the infrastructure, the bandwidth, the capacity, the storage, authentication, attacks, and so on.

Leo: Focus on your strengths, in other words.

Steve: Yes. And it's worth noting that, as trends go, this is definitely a trend. I'll remind everyone that about three and a half years ago, when I was participating in that DigiCert customer advisory board meeting in Utah, and I casually referred to my server rack at Level 3, everyone looked at me like I had just dropped an F-bomb on the Disney Channel. And I said, "What?" And one of the guys said, "Steve, no one does their own hardware anymore." And at that point I thought it best not to mention that I also code in assembly language.

Leo: It's actually an interesting thing. I've been thinking about this because a lot of people, well, a good example is in the password management sphere. There are a certain number of more sophisticated users, probably listeners to this show, who say, oh, no, no, no, I don't want to trust my password vault to LastPass or 1Password or any centralized thing. I'm going to have my own password vault. And while on the one hand I understand, I mean, you're certainly eliminating a target because everybody knows there's a bunch of vaults at LastPass's storage, wherever that is. But on the other hand, you now have to attain the level of professional security that LastPass or 1Password presumably has protecting the vaults. You're assuming the security risk. Just sticking it on Dropbox, I don't know if that's more secure.

So it's an interesting question and tradeoff. And I often tell people, I trust LastPass, or Bitwarden because you can do that yourself, you can self-host Bitwarden. But I trust them to do a better job than I'm going to do. It's their full-time job. Same thing with GitHub. It's their full-time job; right?

Steve: Yes. And, you know, I think certainly in the case of these guys it makes sense. But one thing we have to remember or recognize is that inherently this approach, this consolidation, which is sort of what we're talking about, puts a lot of eggs into many fewer baskets. This makes the care and handling of those baskets far more important than ever.

Leo: Critical, yeah.

Steve: We do see reports, and you were talking about it, and I hear about it on the other podcasts, of spotty outages of major services that transiently bring down all users of the affected service at once. And although I haven't mentioned it before, one of the more notable recent victims of a ransomware attack was one of the largest managed service providers who's been hit with a $20 million ransom demand.

So this sort of consolidation is more cost effective overall. But I think we need to appreciate that it also creates an inherently more fragile solution. This consolidation and

refactoring of function and responsibility is clearly going to happen. And a school district can give its students a day or two or a week off if their informatics systems go down. But mission-critical environments like a hospital might not be able to withstand transient outages. So it needs to be, just as you said, Leo, it's a tradeoff.

Leo: Yeah, you need to understand the tradeoff. It's just not inherently better.

Steve: Right, right.

Leo: And maybe it is. But the burden now is on you. If you're going to host your own Git repository, then the burden is on you to keep it secure. And apparently they weren't able to.

Steve: Well, and I think on balance the industry owes this jokester its thanks.

Leo: Good point.

Steve: He or she made PHP more secure and more securable as a consequence.

Leo: And didn't actually do anything malicious.

Steve: Nope.

Leo: Just called attention to the fact that he could have.

Steve: Yes, exactly.

Leo: Yeah. So he in a way did us all a favor, yeah. Because PHP is just everywhere.

Steve: Yeah, he dropped a big blurb, something you could not fail to notice, that didn't pass any smell test. And everyone's like, whoa. So what the GitHub guys were upset about, I'm sorry, what the PHP guys were upset about was that this happened to them. What the tech press thought was important was, oh, my god, a backdoor was inserted.

Leo: Right.

Steve: And it's like, no. This was a good thing, folks, not a bad thing.

Leo: Yeah. I understand the reaction, though. Randal Schwartz got arrested because he was working at, I won't name the name of the company, working at a company and found a flaw. And he didn't exploit it. He told them about it. And they fired him and got him arrested because they said, "You're hacking our system." And I think

this is not that unusual, where there's a gray line. You're not supposed to be nosing around in there. But if you find a security flaw, I think you're duty-bound to tell the company you found it. This is a good way to do it anonymously without getting in trouble. Who knows. For all we know it could have been Rasmus's son or somebody like that that did it. It's a good thing. There's somebody saying he's not a white hat, not a gray hat, not a black hat. He's a clown hat hacker. Okay.

Steve: Okay. And speaking of Rasmus, who that Rasmus is the father of PHP...

Leo: Lerdorf, yeah.

Steve: My Rasmus is a PHP guru. Leo, you've got a browser in front of you.

Leo: Yes.

Steve: You need to go to www.hiveworkshop.com.

Leo: Ooh.

Steve: That is Rasmus's work.

Leo: This is the guy who does XenForo.

Steve: Well, no. So this is the guy who is a listener of ours.

Leo: Oh, he's our listener.

Steve: Who when I was saying that I was scratching my head about how am I going to integrate SQRL with XenForo, which is written in PHP...

Leo: Right, right.

Steve: He said, "I use XenForo. I'd be happy to help."

Leo: Nice. So this is a XenForo forum, but...

Steve: Believe it or not, that is the most crazy, heavily reskinned, I mean, it's unrecognizable.

Leo: For fans of World of Warcraft. That's cool.

Steve: Yes. It bills itself, HiveWorkshop.com, the No. 1 Largest Warcraft 3, whatever this is, Reforged Modding Community. And I have no idea what that means. But it is a tour de force in PHP-based CSS and HTML reskinning. So, I mean, I can barely see XenForo as I know it under there.

Leo: Yeah, yeah.

Steve: But, wow.

Leo: That's good. That's funny.

Steve: Yeah, great graphics and performance. He moved to the latest XenForo and is apparently cleaning up some little debris. But anyway, I just wanted to give him a shout-out because he's been a big help to us and to the podcast.

Leo: Thank you, Rasmus. HiveWorkshop.com. You get a little plug. How about that? And you get a little respite for a week. That concludes this thrilling, gripping edition of Security Now!: GIT Me Some PHP. Our own personal joker titled this one. You'll find Steve at his website, GRC.com. That's where of course SpinRite lives, the world's best hard drive recovery and maintenance utility. 6.1's coming. Steve's getting there, making some real good progress.

Steve: I'm on it.

Leo: If you buy 6.0 now, you'll get a free upgrade to 6.1. More importantly, you'll get to participate in the development of it. Everybody, if you have a - and I keep saying if you have a hard drive you should have SpinRite. But because now it works so well and does so much on SSDs, if you have any drive, you need SpinRite. I'm getting a new system. I'm going to be getting my SpinRite out to work on the M.2 SSD in it, and it has a spinning drive in it, too.

Steve: It will drive you happy.

Leo: So SpinRite before you - that should be your motto. SpinRite before you go. You'll also find 16Kb versions of this show for the bandwidth-impaired, handwritten human-written transcriptions of every word. Thank you, Elaine. You'll also find 64Kb audio there. There's a feedback form on the website at GRC.com/feedback. There's also a lot of free stuff, including ShieldsUP!, which is really the premier router testing platform. Any time you install a new router, you should go to ShieldsUP!. He's also got a lot of other interesting stuff. It's a rabbit hole you can go down and spend some time. GRC.com. He also is on the Twitter at @SGgrc. You can leave him a DM there. His DMs are open. Slide into Steve's DMs.

We have copies of the show at our website, of course, as with all our shows, 64Kb audio plus video. For some reason we shoot video of it. That's all at TWiT.tv/sn. If you're watching us do it live, we do it live right after MacBreak Weekly of a Tuesday. Usually it's around 1:30 to 2:00 p.m. Pacific. That'd be 4:30 or 5:00 p.m. Eastern,

20:30 to 21:00 UTC. Just, you know what, we're on all day. Go to TWiT.tv/live. There's a live video stream and live audio stream. You can check that out.

While you're doing that, chat with our chatroom. They're watching live, too: irc.twit.tv. You can also comment asynchronously, if you listen to the podcast, at Steve's forums at GRC.com. We also have forums at twit.community, and we have a Mastodon instance. That's the Twitter clone that's federated. It's really cool. We now have enough, I think, critical mass, more than a thousand users, so it's fun. It's perking up. That's twit.social. You're more than welcome to join.

I think, though, if I might, I'd like to encourage you, there's lots of ways to watch the show. There's even a YouTube channel. Get a podcast program and subscribe. That way you'll get it automatically. You won't have to worry about missing an episode. And if you would, if they allow reviews in that podcast player, please give us a nice review. Five stars would be more than welcome. Steve, thank you very much. Have a great evening. Is there an Italian dinner in your forecast for this week?

Steve: Oh, yeah. Steak tonight, Italian on Sunday.

Leo: Steve's fully vaccinated, and he's living it up.

Steve: Ah, yeah.

Leo: I'm right after you, Steve. Have a great week. We'll see you next time on Security Now!.

Steve: Bye.