Out With the Old

Description: This week we address critical updates for Firefox and all Chromium-based browsers and a potentially unwelcome, but reversible, change coming to Firefox. We look at another new tactic being employed by ransomware gangs; an update on ransomware's profitability; a bogus-seeming announcement from Intel during yesterday's CES; and the first use, on this podcast, of the term "teledildonics." Following that, we have some residual SolarWinds news, the formation of a security screw-up crisis management group, news of the inevitable attacks on Zyxel users, the mass exodus from WhatsApp following their plans to force all metadata sharing, and a sci-fi note about "The Expanse." Then, inspired by the amazing amount of old code I have rediscovered inside SpinRite, I will take our listeners back to the roaring '80s with a look at how far we have come from DOS v3.3, whose maximum partition size was 33.5 megabytes.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-801.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-801-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here, and we have lots to talk about. Backspace is back, baby. Steve will explain. We'll also talk about Intel's CES announcement claiming their new processor can detect ransomware. Can it? Plus Steve's got a new security scale. He calls it the "Cheeto scale." It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 801, recorded Tuesday, January 12th, 2021: Out With the Old.

It's time for Security Now!, the show where we cover your security online, your privacy, how things work, how bad it's getting, how good it's getting, all that stuff with this guy right here, live long and prosper, Steve Gibson. Wait a minute. I've got to put the thumb out; right? That's the key.

Steve Gibson: That's right. Thumb out. Oh, Leo.

Leo: I can't do it with my left hand.

Steve: Do I have a show for us today.

Leo: What's the topic?

Steve: Well, the title is "Out With the Old," which was inspired after I got back into SpinRite 6's source code and realized how much crap was in there to deal with the things that we were having to deal with back in the 1980s and '90s at the beginning of the PC era. And so first it was just going to be kind of like, you know, I'll just mention it. But I just think our listeners will really find it interesting. The old-timers will be going, oh, I remember when DOS had a 33.5MB partition size limit, believe it or not. And younger viewers are going to be like, what?

So anyway, we've got so much to talk about. We have critical updates for Firefox and all of the Chromium-based browsers, and a potentially unwelcome, but reversible, change coming to Firefox. We're going to look at another new tactic being employed by the ransomware gangs, or at least one of them; an update on ransomware's profitability; a bogus-seeming announcement from Intel which was made yesterday during their CES announcement of the 11th Gen Core processors. Well, I did dig in to see if there was anything to it, and now our listeners will...

Leo: Oh, I can't wait.

Steve: ...be left with no question about how I feel. And we have the first use on this podcast of the term "teledildonics."

Leo: Teledildonics. Get it right.

Steve: Dildonics. Oh, you're right. Sorry, Leo. Teledildonics.

Leo: As anybody knows.

Steve: Yes. Following that, we have some residual SolarWinds news, the formation of a security screw-up crisis management group, news of the inevitable attacks on Zyxel users, the mass exodus from WhatsApp following their plans to force all metadata sharing for users of their platform, a quick sci-fi note about "The Expanse." And then, as I said, inspired by the amazing amount of old code I have rediscovered inside SpinRite, I'm going to take our listeners back to the Roaring '80s with a look at how far we have come from DOS 3.3. Oh, and we do have a Picture of the Week for the ages. This is just - this is so good. It came without caption, and I thought, that's a Cheeto. So, yeah.

Leo: That could mean a lot of things. Let's see. I've got my T-bar stuck. Wait a minute. All right, good. All right. Unstuck the T-bar. We've come a long way, baby. That's the name of the subtitle on this show. All right, Steve. I'm ready with the Picture of the Week. It does look like a Cheeto.

Steve: This is wonderful. What we have is an attempt to show by clear visual analogy the effect of security when your username is admin and your password is admin. In other words, it's probably what the default was for the device that came with it. Versus a username of KoLpVXriw. That's just the username.

Leo: That's interesting. Do you agree with that? Because most places use your email as the password. It would be, you know...

Steve: Oh, yeah, much better.

Leo: Why use something easily deduced, yeah.

Steve: Yeah, exactly. So yeah, not using your email is definitely better. Of course they do that because when you cannot remember that your password was set to I*$j">?ui$5, they need to be able to email you, to your username, a password recovery link.

Okay. So by way of visual analogy, I'm sure our users are familiar with this very simple slider kind of door lock where there's a slider with a knob that rides in a slot. And in order to lock the door you slide it over, and the end of it goes into a hole cut in a plate. It's like a very simple door lock. Well, that would be too secure if your username was admin and your password was admin. So in this picture that sliding piece has been removed, and a Cheeto has been inserted.

Leo: But Steve, it's a crunchy, flaming hot Cheeto. Not your average Cheeto.

Steve: Better, it's a long, thin Cheeto, yes, in order to perform its task. But it is probably a very good analogy for the equivalent security offered by leaving your username and password both set to their default. And by comparison, over on the right side, where it takes a minute to enunciate the username and password, we've got multiple, redundant, all in parallel, different technologies. That door might as well not exist. It ought to just be a wall that runs right across there.

So anyway, this came from a Twitter follower. Thank you so much for this. I happened to dig back into some of his earlier DMs to me, and I found it. I said, oh, this is too wonderful. Yes, because the slider would be too secure. If it's admin/admin, you need to put a Cheeto in there, Leo. And then it's about the right amount of security.

So Firefox and Chromium have both already been patched for critical updates that were found in them. Of course the two most important Chromium-based browsers are Chrome and Edge, but also pretty much everything else is Chromium now except for Safari. They all required, and it's nice to be able to use the past tense, updating to remove critical remote system takeover bugs. So these were bad. And, you know, this would be much bigger news if all of our browsers did not enjoy near real-time updating. And in fact a bit later we're going to see what's underway now with those Zyxel security endpoints we discussed last week, which do not auto-update.

But in the case of Firefox, this is a 2020 CVE, 16044, carrying the title "Use-after-free write when handling a malicious COOKIE-ECHO SCTP chunk." And the description explains that a malicious peer could have modified a COOKIE-ECHO chunk in an S - it's hard for me to pronounce this because, I don't know, SCTP doesn't roll off the tongue - an SCTP packet in a way that potentially resulted in a use-after-free. They said: "We presume that with enough effort it could have been exploited to run arbitrary code."

Now, we don't often talk about SCTP because I have a hard time pronouncing it. But it is the Stream Control Transmission Protocol, kind of a hybrid of UDP and TCP, and it was originally intended for use within the telecommunications Signaling System 7, SS7, that whole mess. And being a hybrid of UDP and TCP it does run over UDP, so that's its underlying carrier envelope. But it still provides TCP's fault tolerance, reliable delivery, and in-sequence transport of messages, all which, as we know, are missing from UDP, which is just kind of best effort, and we're not sure when or if or in what order the

packets are going to arrive. And like TCP, and unlike UDP, it also handles congestion control. So basically it's sort of a light TCP.

But unlike either of them, SCTP provides multihoming and redundant paths to increase resilience and reliability. It was standardized by the IETF in RFC 4960, so it's a real thing. And although the first reference implementation appeared in FreeBSD 7, support for it is widely present. In any event, the way Firefox uses it, each received SCTP packet contains a COOKIE chunk to facilitate a corresponding reply from the browser's cookie. A COOKIE-ECHO chunk is a snippet of data sent during the initialization of the SCTP connection with the browser. And as Mozilla said, the implementation had a flaw that, given sufficient motivation, could be remotely abused.

Mozilla didn't credit anyone with the discovery. We're not sure where it came from. Nor did they state whether it was actively being exploited in the wild. Maybe they found it being abused and thought, ooh, let's fix this now. But given that our browsers are strict clients of servers, although Mozilla said nothing more, it sounds as though the perpetration of this attack would have required an evil server to send the malicious COOKIE-ECHO chunk back to the browser. This means that it would have required a watering hole attack, drawing browsers, presumably a specific targeted individual, to a malicious site where this vulnerability would have been exploited. But the point being that it's not just something you could get easily. In any event, thanks to auto-updating, we Firefox users are protected.

On the Chromium side, the researchers at Tenable, who found and responsibly reported the bug to Google, believe that it's bad enough to rank it as critical, although both Google and Microsoft classify it as only high in severity. And it's unclear what the back story is on this one, too. It's an out-of-bounds bug that was originally found and reported by Tencent Security Lab researcher Bohan Liu, who received credit for his work. But the bug dates back to a Chrome for Android security bulletin which Google published in October of 2020, when it was also classified as only high severity after its discovery the month before, in September.

So it's a flaw in Chromium's V8 JavaScript and WebAssembly engine. At the time, the bug was also classified as high severity. It's identified as an out-of-bounds write in V8. And today, neither Microsoft nor Google explain why the October 2020 CVE, which is 15995 bug, has popped up again and apparently was just recently in need of fixing, enough that all of our browsers got revved, all the Chromium browsers. But in any event, that was one of an additional 12 Chromium bugs that were reported by Google with, not surprisingly, a one-for-one duplication echoed by Microsoft.

And it's occurred to me that whoever it is over at Microsoft whose job it is to report bugs in Edge now has it very easy. They copy, and they paste. They copy, and they paste. Then they go to lunch. Because just Chromium bugs is all they're dealing with. The majority of the bugs, these 13 in total, were rated high severity and were predominantly use-after-free bugs. And they earned their respective discoverers or teams $20,000 each. So, cool to report things responsibly. And as we've said, if you're good at doing this, you could maybe make it a career.

Unlike the SCTP flaw that was fixed in Mozilla, which due to its nature requires a watering hole-style attack, as I mentioned, vulnerabilities in the Chromium V8 engine are directly exploitable by code that's downloaded from any website, including from an instance of malvertising, even on a highly reputable site. So, good that these things got found and removed, found by the responsible reporters and then quickly revved. And again, browsers, I guess they weren't first; right? Well, maybe they are because they're updating even more frequently than Windows.

But this notion of anything connected needing to take responsibility for its own security, that's clearly the model of the future. I would argue that the browsers are the best at that because they're just doing it all the time. Windows, due to the practical problems of breaking it which cause IT people to lose their minds - it was the case once upon a time, right, Leo, that Windows would just be pushing updates at any time, and that was causing such trouble that they said, okay, fine, we'll aggregate them.

Leo: They consolidated it, yeah.

Steve: And only do them monthly.

Leo: That was mostly for IT departments because individuals probably don't care as much, but IT departments really don't like updates. So it's better if it's once a month.

Steve: Yeah. So the question is, and this is a feature we're losing in Firefox, to change the page or stay where you are? Seven years ago Blair McBride, over at Mozilla, opened a question about what Firefox's Backspace key should do. He opened a Mozilla bug report, although it's not a bug, but the Bugzilla system is just the common way they manage all issues of any sort.

And so seven years ago he wrote: "Pressing Backspace does different things depending upon where the cursor is. If it's in a text field, not surprisingly, it deletes the character to the left. If it's not in a text input field, it's the same as hitting the Back button. Whether to keep this behavior has been argued" - and then he has in initial caps - "For A Very Long Time. It's confusing for many people," he suggests, "but we've assumed it would break muscle memory for many people. However, the muscle memory argument was mostly an assumption. As far as I know, we didn't have useful usage data until now." He says: "So now we have, or can get, useful usage data to either back up the muscle memory argument, or to squash it once and for all and remove Backspace as a shortcut for navigation."

Okay. This is seven years ago. "During the past seven years there has been a long and winding and storied thread accruing comments, observations, and opinions" - and lots of feelings - "until 12 days ago the issue was declared resolved, and a long outstanding question for Firefox's handling of the Backspace key was at long last answered," or at least resolved.

They wrote: "We should not overload common user actions." And for those who are not familiar, "overload" is a term that comes out of object orientation where you can put more functionality into an existing expression which then uses the context of the expression to determine what it does. It's kind of cool, but it can also be confusing. And that's exactly the problem here. We're talking about overloading common user actions, meaning that the Backspace key would be context dependent.

Anyway, they said: "With Backspace, it behaves as you would expect in one context, deleting text; but in another subtle context, text box not selected, it has a destructive action." Now, the reason they call it "destructive" is maybe you were filling in data in a form, and you had a lot, you know, you'd worked at it for a few minutes. And then somehow the insertion point was not in the form, but you didn't realize that, and you hit Backspace, intending to remove a character that wasn't going to get removed. And that was interpreted by Firefox as hitting the Back button, thus clearing, stepping you back

before you received the form, losing all of that work. Thus they call it "destructive." That's what they mean.

So they said: "This creates a high likelihood scenario for a common user behavior, deleting text, to trigger a less likely user behavior, which is using a keyboard shortcut to navigate back." They said: "This investment resolves the overloaded function and closes the gap on user behavior," which is a strange way of putting it. But, they said: "Firefox is the only browser using the Backspace key" - meaning today - "as a keyboard shortcut for navigating to the previous page on a tab. This was originally implemented to follow IE's behavior. We don't go to the previous page if you use the shortcut inside a text input. There are several reasons that make a good case for removal of the shortcut."

And, let's see, the argument to retain Backspace keyboard shortcut for muscle memory for users migrating to Firefox or using Firefox along with another browser does not apply anymore. Edge has now replaced IE as a default Windows browser without a similar shortcut. And they mention that Alt + Left Arrow is the only keyboard shortcut now available for this. Chrome also only implements Alt + Left Arrow because, again, Chromium is common to both browsers. And, they said, Chrome had this shortcut and removed it because of user data loss risks.

So they finish: "The Backspace key shortcut on Firefox is by far the keyboard shortcut with highest usage, with 40 million" - and they use the abbreviation MAU (Monthly Active Users) - "which is well above Find in Page," which actually I use a lot, that has 16 million monthly active users, "or Page Reload," which is 15 million monthly active users. They said: "This raises concerns that our users are suffering usability issues and data loss issues from hitting this keyboard shortcut by mistake." In other words, they can't determine whether it was deliberate or a mistake. But they're noticing it's getting hit a lot, 40 million more than search in a page or reload the page.

So anyway, but because Backspace is so popular, and this podcast is a favorite among the more technically oriented, I would not be surprised to learn that a good percentage of those 40 million monthly active users who are using Backspace to deliberately go back to the previous page might be listening to this podcast and suddenly be confused or annoyed to discover that its use for that purpose has finally come to an end. We're currently at Firefox 84, that is, after those critical updates that I just mentioned. That brought us to 84. The change has been merged, this change in the handling of Backspace has been merged into the Firefox Nightly 86 builds. So we can expect to see Backspace's behavior being neutered once 86 makes it to the main release channel.

The good news is that the Boolean setting for it can be turned back on. As usual, you put about:config into the URL of your Firefox browser and hit Enter. That brings up a bazillion settings. Oh, you now have to agree to accept the awesome responsibility that accompanies making any changes behind the scenes and then search for the string "browser.backspace." One item will be found. It's browser.backspace underscore something, I don't even remember now what it was. But just search for "browser.backspace," and you'll find it.

The option in my Firefox 84 was Boolean "0," and that does give you page-changing Backspace behavior that we've always had. I experimented with it. I set it to "1" and then restarted Firefox, assuming that I had to. I didn't actually check that. And that, sure enough, causes Backspace to be ignored when you're not entering text. And so the point is this is already in there, and it's in about:config, one of Firefox's bazillion settings. So this suggests that, with release 86, they're going to simply flip that existing zero to a one in order to flip the default behavior of Firefox. And so anybody who will miss that can easily go in and turn it back to zero, as it is today.

A couple notes about ransomware. Security researchers following the flow of bitcoin transactions from the victims of Ryuk's ransomware into the bad guys' pockets estimate that the Ryuk operation has netted at least, that is, they've been able to track at least $150 million. They discovered that Ryuk's operators primarily use two legitimate cryptocurrency exchanges to cache out the bitcoin from paying victims into fiat money, regular money.

Two threat intelligence companies, Advanced Intelligence and HYAS, tracked 61 bitcoin wallets attributed to the Ryuk malware enterprise. They discovered that the cryptocurrency moves from an intermediary to the - it's H-U-O-B-I, Huobi maybe, and Binance exchanges. When Ryuk victims pay the ransom, the money first passes through a broker, who passes it to the malware operators. The money then goes through a laundering service before getting to legitimate cryptocurrency exchanges or being used to pay for criminal services on underground markets.

The researchers explained that in addition to the large and well-established Huobi and Binance exchanges, there are significant flows of cryptocurrency to a collection of addresses that are too small to be an established exchange and probably represent a crime service that exchanges the cryptocurrency for local currency or another digital currency. One of the largest transactions involving a Ryuk wallet found during this investigation was more than $5 million in a single transaction, or 365 bitcoin at the time. But that payout was not the largest ransom ever paid. In an earlier report, Advanced Intelligence said that the largest payment confirmed to these attackers was 2,200 bitcoin, which converted to $34 million at the time.

Leo: Jesus. Holy cow.

Steve: $34 million, Leo. You could buy yourself a whole new, like, everything. Network and servers, I mean, wouldn't you love to know what company thought, well, yeah, we should pay that?

Leo: Good lord, yeah, wow.

Steve: The average ransom value received by Ryuk is 48 bitcoins. So they're making money.

Leo: Even then, that's a lot, yeah.

Steve: Yeah, even that. Another highly profitable ransomware gang we've talked about often is REvil, or Sodinokibi, who announced through a public-facing representative that they made $100 million in one year from extorting their victims. They said that their goal was to make $2 billion. As we know, their affiliate-based program does potentially allow them to grow and expand their rate of incursion. A few months ago we noted that they had ended their "We have all the people we need for now" status and are again seeking to recruit new affiliates into the fold. So they're in a growth phase at the moment.

Okay, now, yesterday during the 2021 CES, right, the Consumer Electronics Show, with great fanfare, Intel announced that their 11th-generation Core vPro CPUs will have support for their boot-protecting Hardware Shield and something known as Threat Detection Technology (TDT). This Threat Detection Technology, they claim, will be able to "detect ransomware attacks at the hardware silicon level, many layers below antivirus

software." Now, okay, naturally that was of interest to me. So I spent some time digging a bit deeper into these claims. And frankly, it's nonsense.

Leo: Yeah, sounds like nonsense.

Steve: It appears to be pure marketing hype. The press - I know. The press release said: "Intel TDT uses a combination of CPU telemetry and ML" - of course that's now one of the buzzwords, Machine Learning - "heuristics to detect attack behavior. It detects ransomware and other threats that leave a footprint on Intel's CPU performance monitoring unit (PMU)." They said: "The Intel PMU sits beneath" - they ought to just call it the PU - "sits beneath applications, the OS, and virtualization layers on the system and delivers a more accurate representation of active threats" - and I'm shaking my head while I'm reading this - "system-wide." They said: "As threats are detected in real-time, Intel TDT sends a high-fidelity signal that can trigger remediation workflows in the security vendor's code." Okay. What we have there is an example of some creative writing.

Leo: Yeah, no kidding.

Steve: The biggest malware threat faced by the enterprise we know today is ransomware. And nothing has been able to slow it down so far. But wait. Intel has a new chip, and it has a hardware ransomware detector built-in.

Leo: Wow.

Steve: That can finally, yeah, it can finally put an end to this scourge.

Leo: Woohoo.

Steve: So quickly cancel your short position on Intel stock.

Leo: Oh, yeah. Oh, yeah.

Steve: Yeah, and instead buy as much as you can, kids. Intel's future is once again bright. Except of course none of that is true. The marketing guys apparently met with the engineering guys, and they asked: "Okay, so the Gen 11 vPro processors have this, what, you call it a PMU? What's it good for? How can we sell that?" And the engineering guys scratched their heads a bit and said: "Well, in theory, since the PMU is seeing everything that's going on at the processor core level, it's impossible to hide anything from it. Really, truly impossible. So that means that, if something malicious was going on anywhere in the system, even if it was tucked away inside a virtual machine, it would have to affect the PMU." Whereupon the marketing guys interrupted and said: "Hold on. Wait a minute. Does that mean that the PMU could detect ransomware?" And the engineers said: "Well...." And the marketing guys said: "Perfect. Now we can sell the crap out of this thing."

To give you a feel for the flavor of this nonsense, Stephanie Hallford, Intel's Client Computing Group Vice President and General Manager of Business Client Platforms, was quoted to say - that doesn't fit on her business card, by the way, it has to wrap around the back: "Ransomware was a top security threat in 2020. Software alone is not enough to protect against ongoing threats. Our new 11th Gen Core vPro mobile platform provides the industry's first silicon-enabled threat detection capability" - wow, that sounds great - "delivering the much-needed hardware-based protection against these types of attacks. Together with Cybereason's multi-layered protection, businesses will have full-stack visibility" - because, again, that's another buzzword. You've got to have "full-stack" in there somewhere - "from CPU telemetry to help prevent ransomware from evading traditional signature-based defenses." And then the press release noted that although Cybereason will be the first to support detecting ransomware using hardware indicators, other security vendors will most likely tap into it in the future.

Okay. So I wanted to provide some preemptive context for this announcement in case any of our listeners encounter the news of an Intel breakthrough in ransomware prevention. It didn't happen. I will double-dip guarantee you that a system that's full armed with this nonsense is every bit as vulnerable as the systems we have today. Could it possibly be of some value? Sure. But only if you knew precisely what to look for in advance, and only if regular system activity didn't constantly throw up false positives.

At first blush, it sounds amazing that by being down so low in the silicon there's no way for the actions of ransomware to avoid detection. That would be true. But the trouble with being so low down in the silicon is that all context is lost. Down there, you have no idea what's going on up above. You see registers being loaded and unloaded. Math is happening. Jumps are being taken or not. Memory is being read and written. Subroutine calls are being made and returned from, and transitions are being made among various OS privilege rings.

But it's all completely anonymous. There's absolutely no context. Everything, everyone is reading and writing registers, doing math, retrieving and storing to memory, making OS calls. So there's no way to know what any of that actually means down there. Maybe a malware prevention system operating up at the application layer could augment everything it already knows with the information that will someday be available from an Intel 11th-generation vPro processor which contains the optional extra cost PMU. Maybe. But even getting to maybe seems uncertain to me. So, yeah.

Leo: So the PMU just really is seeing microcode or assembly language level stuff.

Steve: Yeah. It's just seeing instructions. It's just seeing instructions.

Leo: I guess if there were a typical signature for ransomware, an instruction only ransomware used or something like that, you might say, well, that's suspicious. But I can't imagine it looks that different.

Steve: Unfortunately, there ain't no such thing. One of the problems is ransomware looks like just another application.

Leo: Yeah. It's encryption.

Steve: It's running, and it's reading and writing things and doing stuff.

Leo: Yeah. Wow.

Steve: So Leo, we now have the strange case of the male chastity cage.

Leo: Oh, about time. I can think of some people who should wear this.

Steve: Yeah. On the lighter side, I suppose, unless you're unfortunate enough to be wearing one, we'll remember those bizarre IoT chastity devices for men that we briefly covered some time ago. It's the Cellmate, I guess as in cell phone, it's the Cellmate Chastity Cage, a bargain at $169. That's for the small model.

Leo: Oh, I need a large, I'm sorry, I can't...

Steve: As I've taken to say more recently, "What could possibly go wrong?" Although perhaps "What is wrong with this picture?" would be more fitting in this instance. So it should come as no surprise to anyone that the software behind these devices - would that be firmware? Anyway, behind these devices...

Leo: It's all hardware, baby.

Steve: Yeah, the hardware. But hopefully not while you're wearing this thing.

Leo: Ow.

Steve: ...was not of the highest grade. Last October, researchers at Pen Test Partners published details about a serious vulnerability that allowed a remote attacker to take control of any Cellmate device. They began their disclosure with the observation: "In this research, it helps to have an open mind and to not be judgmental."

Leo: I'd rather have an open chastity belt.

Steve: Yeah. In an effort to keep, I guess, it a bit tongue-in-cheek, they obtained a subdomain of the .GS top-level domain which allowed them to post their serious and authentic security research at the easily remembered domain, InternetofDongs.

Leo: Oh, geez. Oh, god, I'm so sorry. So sorry.

Steve: In their TL;DR they summarize the situation in a series of bullet points: Smart Bluetooth male chastity lock, designed for user to give remote control to a trusted third party using mobile app/API. Multiple API flaws meant anyone could remotely lock all devices...

Leo: Oh, no.

Steve: ...and prevent users from releasing themselves.

Leo: Oh, no.

Steve: Removal then requires an angle grinder or similar, used in close proximity to delicate and sensitive areas. Precise user location data is also leaked by the API, including personal information and private chats. You know, "Get me out of this thing." Vendor initially responsive, then missed three remediation deadlines they set themselves over a six-month period, finally refusing to interact any further, even though majority of issues were resolved in migration to v2 API, yet API v1 inexcusably remained available. And I should say "remains [current tense, present tense] available."

The researchers wrote: "We are not in the business of kink shaming. People should be able to use these devices safely and securely without the risk of sensitive personal data being leaked. The security of the teledildonics field is interesting in its own right."

Leo: Oh, yes.

Steve: "It's worth noting that sales of smart adult toys has risen significantly during the recent lockdown."

Leo: Well, that makes sense, yes.

Steve: And I suppose this brings new meaning to the term "lockdown," Leo.

Leo: And by the way, on a related note, I want to announce a new sponsor for the show, ManEscaped, which is software to help you get out of your chastity belt, if you inadvertently get locked in. ManEscaped.com. Okay, go ahead. That was ad number two.

Steve: Okay. So in describing the risk to users, they wrote: "We discovered that remote attackers could prevent the Bluetooth lock from being opened, permanently locking the user in the device. There is no physical unlock. The tube is locked onto a ring worn around the base of the genitals, making things inaccessible. An angle grinder or other suitable heavy tool would be required to cut the wearer free. The user's location, their plaintext password, and other personal data was also leaked, without need for authentication, by the API."

They said: "We had particular problems during the disclosure process, as we would usually ask the vendor to take down a leaky API whilst remediation was being implemented. However, anyone currently using the device when the API was taken down would also be permanently locked in. As you will see in the disclosure timeline at the bottom of this post, some issues were remediated, but others were not. And the vendor simply stopped replying to us, to journalists, and to retailers. Given the trivial nature of finding some of these issues, and that the company is working on another device which

poses even greater potential physical harm," and they said in parens "(an internal chastity device), we felt compelled to publish these findings at this point."

So somewhat predictably following the disclosure, an attacker started targeting Cellmate mobile app users. Thus this is in the ransomware section of the podcast because they were being asked to pay 0.02 bitcoin, around $270, at the time of the attacks in order to have their device released. All of these things have been locked, and users are now having to pay to be released. So thus we witness the rare birth of a new class of ransomware.

So yesterday's post by Kaspersky Labs was titled "Sunburst Backdoor. Code overlaps with Kazuar." It's K-A-Z-U-A-R, a known Russian sourced malware. And in their post, following their de rigueur "What is Sunburst" reminder, as if anyone reading a Kaspersky posting doesn't already know, they introduce us to what they have found.

They wrote: "In a previous blog, we dissected the method used by Sunburst to communicate with its command-and-control (C2) server and the protocol by which victims are upgraded for further exploitation. Similarly, many other security companies published their own analysis of the Sunburst backdoor, various operational details, and how to defend against this attack. Yet, besides some media articles, no solid technical papers have been published that could potentially link it to previously known activity."

They said: "While looking at the Sunburst backdoor, we discovered several features that overlap with a previously identified backdoor known as Kazuar. Kazuar is a .NET backdoor first reported by Palo Alto Networks in 2017. Palo Alto tentatively linked Kazuar to the Turla APT group, although no solid attribution link has been made public. Our own observations indeed confirm that Kazuar was used together with other Turla tools during multiple breaches in past years. A number of unusual, shared features between Sunburst and Kazuar include the victim UID generation algorithm, the sleeping algorithm, and the extensive use of the FNV-1a hash."

Okay. So on the podcast we've never had occasion to talk about the FNV-1 hash because it's not cryptographically secure. FNV are the initials of the hash's inventors - Fowler, Noll, and Vo. What the hash can boast of is its very small implementation size, and its speed, and its fully tunable output length. The hash operates in a very simple loop. The current hash is multiplied by an FNV prime value; then a byte of input is XORed into the result. That's repeated until all of the input has been consumed. And I looked at the actual code implementation in Kaspersky's posting, and it's like, five instructions. So it's going to be screamingly fast. And it provides a way for them to create a very fast lightweight hash.

Anyway, the Kaspersky Report goes on in great detail, showing reverse-engineered code from Sunburst and Kazuar, and specifically addresses the problem and challenge of false-flag deception where attribution is deliberately misdirected. And finally, after a long posting, they conclude, saying: "These code overlaps between Kazuar and Sunburst are interesting and represent the first potential identified link to a previously known malware family. Although the usage of the sleeping algorithm may be too wide, the custom implementation of the FNV-1a hashes and the reuse of the MD5+XOR algorithm in Sunburst are definitely important clues. We should also point out that although similar, the UID calculation subroutine and the FNV-1a hash usage, as well as the sleep loop, are still not 100% identical."

They said: "Possible explanations for these similarities include: Sunburst was developed by the same group as Kazuar. Or the Sunburst developers adopted some ideas or code from Kazuar, without having a direct connection, like they used Kazuar as an inspiration point. Or both groups, Dark Halo/UNC2452 and the group using Kazuar, obtained their malware from the same original source. Or some of the Kazuar developers moved to

another team, taking knowledge and tools with them. And finally, or the Sunburst developers introduced these subtle links as a form of false flag in order to shift blame to another group."

So still, at the moment, now weeks downstream of this, and with no little amount of attention and focus being brought to bear, we do not know which one of these options is true. While Kazuar and Sunburst may be related, the nature of this relation is still not clear. Through further analysis, it's possible that evidence confirming one or several of these points may arise. At the same time, it's also possible that the Sunburst developers were really good at their operational security and didn't make any mistakes, with this link being thereby an elaborate false flag. In any case, this overlap doesn't change much for the defenders. Supply chain attacks, as we know, are some of the most sophisticated types of attacks these days, and they've been successfully used in the past by APT groups.

The Kaspersky guys note: "To limit exposure to supply chain attacks," they said, "we recommend the following: Isolate network management software in separate VLANs. Monitor them separately from the user networks. Limit outgoing Internet connections from servers or appliances that are running third-party software. Implement regular memory dumping and analysis, checking for malicious code running in a decrypted state using a code similarity solution."

And that's actually, that third one is very cool. That's one of the ways that Kaspersky's proprietary tools operate is that we know that polymorphic encryption is used to prevent signature analysis by AV tools. Each instance of the malware is encrypted under a different key so that it looks like completely different random gibberish. And even the decryption algorithm that must be somewhere in the decrypted blob, that's dynamically generated so that it never looks the same twice. But what's common among all of those is, once it's decrypted and running in RAM, it's the same item every single time. It is highly repetitive. But it's only once it's in memory that you're going to see that.

So anyway, as we know, Kaspersky is a Russia-based security firm. They've taken a lot of heat about that over the years. We've touched on that from time to time. Does that color their conclusions? We don't know. Attribution, as we know, in cyberspace is incredibly difficult. And it remains so.

As for the U.S. government, last Tuesday our intelligence and law enforcement agencies here in the U.S., including the FBI, CISA, the ODNI, and the NSA, jointly published a statement. They said, speaking of this massive attack: "This work indicates that an Advanced Persistent Threat actor, likely Russian in origin, is responsible for most or all of the recently discovered ongoing cyber compromises of both government and non-governmental networks."

So, okay. The press jumped on this as the U.S. government formally accuses Russia. And it's like, okay, wait. They said "likely." And, you know, frankly, "likely Russian" certainly falls far short of being definitive. And at this point it seems that the inherent difficulty of attribution in cyberspace is what rules the day. And I'm, frankly, impressed that they're not claiming proof that they don't have. And the problem is it's just difficult to prove. I heard you recently, I think it was on some other podcast, Leo, talking about the problem of attribution in cyber.

Leo: Yeah. Right.

Steve: It's just, you know, it's so possible to set up a false flag, to deliberately bounce IP traffic through someone else or to use someone else's code, which anybody can get by

reverse engineering, in order to set up something that looks like it's the work of those people. So you just don't know.

Leo: Turla is another Russian group, though, thought to be Russian. So Kaspersky wanted to defer responsibility...

Steve: Oh, to really point fingers elsewhere, yeah.

Leo: Yeah. I think they would choose something that's not Russian. So, yeah. I do think, though, that it's completely credible. These guys are so good that they copied or intentionally put those false flags in there. Or maybe they had the source code for Turla and just said, ah, that works well. We'll just borrow that.

Steve: Well, yeah. And you'd have to think also that, as you say, being as good as they are, they would recognize the chance of the connection being made. Or maybe they just don't care.

Leo: Or maybe they don't care, yeah.

Steve: Yeah. You know? We're going to stay hidden as long as we can. Once we're no longer hidden, well, then it doesn't matter. And also, unfortunately, what are you going to do about it?

Leo: Yeah, that's really the big point. So, what are you going to do?

Steve: Yeah, what are you going to do?

Leo: What are you going to do?

Steve: So in SolarWinds news, we also talked about the trouble that the SolarWinds Corporation would likely be facing in the future. Predictably, a class-action lawsuit has been filed by shareholders of SolarWinds' beleaguered stock. The named defendants in the suit are SolarWinds President Kevin Thompson and Chief Financial Officer J. Barton Kalsu. The suit alleges that the executives violated federal securities laws under the Securities Exchange Act of 1934. And embedded in the suit was a bit of information that raised my eyebrows. Among the many other grievances enumerated by the plaintiffs was the fact that the SolarWinds update server used the password "solarwinds123."

Leo: "Password123," I think. Wasn't it password123?

Steve: I think it was solarwinds123.

Leo: Solarwinds123, wow.

Steve: So again, wouldn't be a big stretch. It's not the first thing you would guess; but, wow, it's certainly not uppercase Z`q and so forth.

Leo: It's more on the Cheeto side, yeah.

Steve: Solarwinds123, it's better than a Cheeto, yeah. In fact, Leo, that's got to be one of our new, you know, we have TNO, Trust No One. And now we have "Better than a Cheeto." Or maybe "Beats a Cheeto," yeah.

Okay. So the Krebs Stamos Group. In a bid to manage the security side of the mess created by the SolarWinds hack, SolarWinds has hired a freshly founded consultancy known as the Krebs Stamos Group.

Leo: I like the name.

Steve: I do. And boy...

Leo: Which Krebs is it? Not Brian.

Steve: It's not Brian.

Leo: It's Chris Krebs, yeah, former...

Steve: And what's cool is the URL. .GROUP is a TLD, so they are KS.group.

Leo: Oh, that's a good URL, yeah.

Steve: Krebs Stamos Group. So the front page banner of that page - actually it's not the front page, it's the only page of their new website - declares: "Krebs Stamos Group helps organizations turn their greatest cybersecurity challenges into triumphs." And in the case of SolarWinds, well, good luck with that.

Leo: Can they turn my bitcoin into dollars? That's what I want to know.

Steve: Yeah. And yes, both of those names should be familiar to our listeners. Krebs is not Brian Krebs, it's Chris Krebs, the original and now previous head of the United States CISA, who while still head of CISA had the unwelcome temerity to publicly state that the recent U.S. presidential election was the most secure ever. Whereupon Chris became the previous head of CISA.

And Stamos is of course Alex Stamos, Facebook's former CISO and founder of the Stanford Internet Observatory. As we know, shortly following the multiple Zoom security debacles in early 2020, Alex has been helping Zoom manage their new security challenges. Presumably now this will fall under the Krebs Stamos Group umbrella. And it's going to be interesting to see where this duo pops up in the future. As the industry's

highest profile security disaster response group, I have the feeling that we're going to be seeing more of them.

Maybe, speaking of which, maybe Zyxel - is that how you pronounce it? Zyxel?

Leo: I always say "Zycel." I don't know if that's accurate or not.

Steve: Zyxel, yeah, that's sounds...

Leo: Used to have a Zyxel modem, way back when. It was a 56k baud modem.

Steve: Yeah, it was originally really good hardware.

Leo: They were the best modems, yeah.

Steve: Yeah, yeah. So last Tuesday, right while we were talking about the new threat facing the more than 100,000 Zyxel VPN and other security endpoint customers, they've got a whole family of security endpoints, GreyNoise Intelligence was observing that the previously unknown "zyfwp" account name had been added to SSH scanners and was now being actively probed across the Internet.

Leo: Of course. Of course.

Steve: Uh-huh. Why wouldn't you? Johannes Ullrich, of the SANS Internet Storm Center last week said: "Likely due to the holidays, and maybe because the discoverer of the backdoor account did not initially publish the matching password, widespread exploitation via SSH had not started until now. But we are seeing attempts to access our SSH honeypots via these default credentials." Ullrich said the scans started last Monday afternoon, stemming from one IP, 185.153.196.230. And more scans from other IPs, 5.8.16.167 and 45.155.205.86, joined throughout this week, that is, past week.

He said: "The initial IPs scanning for this are all geolocating back to Russia. But other than that, they are not specifically significant. Some of these IPs have been involved in similar Internet-wide scans for vulnerabilities before, so they are likely part of some criminal infrastructure." Of course this backdoor account is now widely public, and so it permits an incredibly serious risk because anyone on the Internet can now use it to create new VPN accounts to give themselves access to internal networks or to port-forward internal services to make them remotely accessible and exploitable. It is looking like it's going to become a horrific disaster. I have a hunch we'll either see a big jump in ransomware attacks, or just news of people becoming victims to this. I don't know, 100,000 of these, and they're in back rooms. They're in closets. And then Zyxel says, oh, be sure to update. Hello, who? Wow.

Meanwhile, up until now, WhatsApp's privacy policy opened with the claim that "Respect for your privacy is coded into our DNA." Remember that?

Leo: Mm-hmm.

Steve: They said: "Since we started WhatsApp, we've aspired to build our services with a set of strong privacy principles in mind." Well, that was then. I think it was 2016. The news that the respect for the privacy of WhatsApp's user metadata is being lost has triggered a mass exodus from Facebook's in-house communications platform over to Signal, where no user metadata is ever collected.

Back in 2016, WhatsApp gave its users a one-time ability to opt out of having account data turned over to Facebook. But now their updated privacy policy, taking effect next month, on February 8th, changes that. One month from now, users will no longer have that choice. Some of the data that WhatsApp collects and will be sharing includes users' phone numbers, other people's phone numbers stored in address books, profile names, profile photos, location information, transactions and payment data, device and connection information, status messages, including when a user was last online, and diagnostic data collected from app logs.

And so, yeah, they may not be seeing what you're saying, but they have and will shortly be using every other last scrap of available metadata. Under the new terms, Facebook reserves the right to share collected data across its family of companies. And in some cases, such as when someone uses WhatsApp to interact with third-party businesses, Facebook may also share information with those outside entities. This privacy agreement update is a 180 compared with last year's privacy policy whose enforcement began last July. It states that users are able to choose not to have their WhatsApp account info shared with Facebook to improve the company's ads and products. But under the changes to the policy, users will now be forced to accept sharing their data with Facebook to continue using their account; or, as an alternative, delete their account.

WhatsApp's notification reads: "By tapping AGREE [all caps] you accept the new terms and privacy policy, which take effect on February 8, 2021. After this date, you'll need to accept these updates to continue using WhatsApp." So given the supreme lack of subtlety shown, it really does feel like a bit of a bait and switch.

On the other hand, I mentioned that Signal was experiencing a mass influx of ex-WhatsApp users. It turns out that Signal's infrastructure was for a while a bit unprepared and overwhelmed by this rush. BleepingComputer reported last Friday that Signal had now recovered from sign-up delays which had previously been affecting its user verification process. When setting up Signal for the first time, users must verify their mobile number using verification codes sent by Signal. And Leo, you noted that this is sort of an unfortunate loop that Signal is requiring. There are some other good end-to-end encryption solutions that don't require that.

Anyway, the surge in new users switching to Signal overwhelmed the verification service, causing significant delays across various mobile providers. Of course that's the sort of problem you'd like to have. Signal tweeted on the 7th, when they were finally beginning to recover, that was last Thursday: "Everyone should be able to register without delay again. Thanks to all of the carriers who flipped the right switches so that people can keep switching." I don't know what that means. Maybe they got shut down due to a tweet storm or an attempt to do too many SMS messages in a short period of time.

On the 8th Signal tweeted: "New users in Europe should be able to register without delay again. We will continue to work with carriers to keep delivering a record-high number of Signal verification codes. The Signal service itself is operating normally. Thanks for your patience." Later the same day, on January 8th: "Some new users are reporting that Signal is slow to display their Signal contacts. We're adding more capacity to keep up with all the new people searching for their friends on Signal. Hang tight."

The next day, on the 9th, they tweeted: "Not to sound like a broken record about broken records, but we're aware of the registration delays while creating a new account. Carriers

are making adjustments on their side to keep delivering verification codes as quickly as possible." This must have been a mass exodus. Later on the 9th: "Even though we're still breaking records, verification codes are back in the groove. Delivery delays have been eliminated across multiple cellular providers, so things should be more ASAP when you join the app."

And, finally, on the 10th, which was Sunday, two days ago, they tweeted: "We continue to shatter traffic records and add capacity as more and more people come to terms with how much they dislike Facebook's new terms. If you weren't able to create a new group recently, please try again. New servers are ready to serve you." So, wow.

Leo: That's great. I'm really happy to hear that. I hope people make the shift. Be great.

Steve: Yeah. And I guess certainly Facebook has the right to whatever terms and conditions they want. I'm sure there are lots of users that will still use Facebook because it's, I mean, still use WhatsApp because it's part of the Facebook family of properties, and getting more relevant ads is what they want. So it's like, okay. But lots of users are clearly a little annoyed. And notice that, again, this is not anyone's conversation that is being decrypted. This is just all the metadata. So it does also feel like people are beginning to understand more about the subtleties of the difference between your conversation versus metadata.

Leo: Thanks to Apple really exposing that, yeah.

Steve: Yeah. Yes, and Apple has been moving in the other direction, as they know. They're getting better about requiring apps to be very clear about exactly what they're going to be doing with their users' data.

So I did want to just take a moment to mention "The Expanse" on Amazon Prime. We first talked about "The Expanse" many years ago, when it was - it was initially, for the first three years, it was a Syfy property. And when it had been announced, I remember Mark Thompson telling me that it was coming, and he was excited. He had read...

Leo: You loved the novels, I know.

Steve: Yes. He had read them. I immediately read them because, as we know, the books are always better. Although, boy, I just started rewatching it. I know that you have. We talked about this last week or the week before. Lorrie had never seen it. I was a little worried that it might be a little too out in the weeds for her. But she's tolerating it, at least. And I just have to say I had forgotten how good it was. I mean, it is really good. I mean, you need to really pay attention. There is some problem, like with articulation, where it's like, what did she just say? But it doesn't - that kind of stuff really doesn't matter. And they're speaking in a language that they made up, little snippets here and there which are just to provide some color. And again, not important to the plot.

But for what it's worth, so what happened was, it was incredibly popular on Syfy. And they chose to not continue the series, not because it wasn't super popular and the production values weren't, like, amazingly high, especially for the Syfy channel, but over some contractual problems with distribution. Things got - I don't remember. I don't know

what the details were. But they just said, okay, fine, if that's the way you're going to be, somebody, then we're not going to do it again.

A crowd-funded airplane was financed to fly circles around Amazon headquarters, trying to get Bezos to pick it up. And Jeff did. And what I found interesting was the ratings have...

Leo: I didn't know that. That's why it got picked up. Wow.

Steve: Yeah.

Leo: I had no idea.

Steve: And the ratings have only gone up since then. It is now - the last few seasons are 100% on Rotten Tomatoes. And it's always in the high 9s on IMDB. So anyway, for what it's worth, again, there is a lot of, like, well, in fact, at one point Lorrie said, "I hope this is not going to be constant shooting." And it's anything but. I mean, my concern is that it was a little too much internecine political machinations between Earth and Mars and those out in the Belt, the Belters. But I just think it's wonderful. So just a heads-up to our listeners that - and it's on Amazon Prime.

Leo: Yeah, you probably get it free, yeah.

Steve: You're a Prime user, probably get it for free. And boy. And five seasons now. And I think maybe I saw the first three before, so I'm having no problem rewatching them. As I said, I forgot how really high quality it was. And before long I'll be going past where I read in the novels out into new territory.

Okay. So the title of the podcast is "Out With the Old," inspired by my previous week of digging around inside SpinRite. I've frankly been amazed as I've been encountering and remembering how much functionality I managed to cram into this system over time. And it made me think back. There were so many kludges - my favorite word again - perpetrated upon the industry as our early PCs kept outgrowing their modest beginnings. Of course Gates was famously quoted saying, you know, the previous CPM machines were absolutely limited to 64K of RAM. And Bill was quoted as saying, when the IBM PC came out with 640K, whoo, 10 times as much, "Oh, we'll never need more than that."

Anyway, one of the many problems was that mass storage size grew beyond anyone's wildest expectations. And as we know, in the 40 years since the PC's August 12, 1981 - so Leo, this August 12th the PC will be 40 years old. And we're 40 years older, my friend. But this pace of mass storage growth has never let up. So to remain useful, SpinRite had to deal with every weird thing a system might do. For example, when the total sector count limit was reached, device drivers were created to deliberately misrepresent the size of sectors on the system's media. If you couldn't have more sectors, at least the sectors you could have could be made larger.

Physical sectors transferred from devices, the actual physical devices, have always been 512 bytes. But an intermediate device driver would tell DOS that the mass storage device was using, for example, 4096-byte sectors. So that's what DOS would see. And that's the way it would format the drive. All file system data structures would then occur in what was actually eight physical sector multiples rather than the truth of one physical sector.

And this was fine for DOS since it really didn't care how big sectors were. It believed what it was told.

But of course SpinRite was accessing the drive directly and beneath any device driver. So it needed to juggle this 8-to-1 - and it wasn't always 8-to-1, it could be any multiple of logical versus physical sector reality. And that juggling had to happen for everything SpinRite did. And remember that the very popular DOS at the time, v3.3, DOS 3.3 was like the one we had for a long time at the beginning. It had a - get this - a 32MB limit. 32MB.

Leo: Per file. Per file.

Steve: No, no, no, no. DOS 3.3's entire partition, the C partition.

Leo: Only 32 megs?

Steve: Yes, 32 megs. It was actually 33,554,432 bytes. Where did that come from? It came from DOS's logical sector number, which back then was 16 bits. DOS had a 16-bit sector number.

Leo: Of course, most people didn't have hard drives that big, so it was okay.

Steve: Right. You could get - you could choose. Do you want 5MB or 10MB?

Leo: I remember when I got a 20MB drive. I was so excited.

Steve: Oh, Leo.

Leo: That was that new MFM technology. So 32MB probably at the time seemed fine. We'll never need that much.

Steve: The 20MB was the Seagate ST-225 drive.

Leo: That's right. Boy.

Steve: So 16 bits gives us 65,536; right? 64K in binary sectors. You multiply 65,536 sectors by 512 bytes per sector, and you get 33.55 and some megabytes. That was the largest possible partition that DOS 3.3 could handle. And it's interesting. As I was thinking about all this, while we're talking about PC history, when I encountered that 33.55MB limit, I was reminded of something I wrote long ago. One of the ideas I had when I was writing the InfoWorld TechTalk column was to design what I called "Steve's Dream Machine," where every component was carefully selected for cost-effectiveness and value, like this is the motherboard you want, and exactly why. This is the RAM you want, and exactly why. And so on.

Well, there was a Miniscribe 3650 drive that was 42MB under its rated MFM encoding, but it was high-quality enough to handle RLL's 50% storage increase, bringing it up to 64MB when formatted with an RLL controller, which was - and the Adaptec 2372, that was my choice of the best controller. And it turned out that the drive advertised fewer cylinders than were required to deliver a second partition because, okay, if you have a 42MB drive, but you hook it to an RLL controller, now it's at 64MB. But DOS 3.3's maximum partition size is 33, or 32 binary megabytes. So you've got to have two partitions because you can't just have one. The drive's now too big for the maximum DOS partition.

So it turned out that the drive advertised fewer cylinders than were required to deliver a second partition of the maximum possible size under DOS 3.3. Although they weren't advertised, the required additional cylinders were physically there. And I discovered that they worked. So one of the coolest hacks I used and promoted for Steve's Dream Machine was to format the drive further toward its spindle to obtain additional storage. Then FDISK could create a C and a D that each contained the absolute maximum size of a DOS partition, that 33-plus million sectors.

And, you know, it was a gimmick. But it was fun, and the concept was quite popular back then. And I thought, I wonder if this is still online? Because I know that a lot of InfoWorld ended up getting put online. I actually found, if anyone is interested, it is sort of a blast from the past, I found the text of that original InfoWorld column online. It was titled "A Hard Disk Drive for Steve's Dream Machine." And I excerpted the relevant portion.

I said: "The Miniscribe 3650 is not quite officially RLL certified, though I hear rumors that it's about to be, simply because it works so well. I've tested many of them myself, and the bright boys at Northgate Computer Systems who turned me on to this drive in the first place are shipping thousands with RLL controllers in their 286 AT compatibles. They've had no problems. I'm quite comfortable with the 3650 and RLL encoding.

"Finally, the 3650 is rated as having 809 cylinders, though it actually has 852. I've been low-level formatting mine out to 842 cylinders. Then, under DOS 3.3 with RLL encoding, you get two maximum size 33.4MB DOS partitions! They couldn't be any bigger! Sixty-seven fast megabytes in an inexpensive half-height drive is hard to beat."

Leo: Stand back. Whoo. Whew. Wow.

Steve: Wow.

Leo: Somebody typed that in, by the way. That's hand-typed from your column.

Steve: Hand-transcribed from the column.

Leo: That's hysterical.

Steve: So anyway, as I was saying, DOS back then was designed to manage a maximum of 64K sectors. But if you needed more storage, you could add a device driver to lie about the size of each of those sectors to obtain many times the storage. And I wrote this as I was - this just came to mind as I was writing this. I think there was a company called OnTrack that bundled its driver with many hard drives at the time.

Leo: Yeah, I remember it, yeah.

Steve: Which is a device driver that did that with many hard drives. Because, you know, if you were a Seagate or a Western Digital or a Miniscribe or anybody who was making larger hard drives, yet DOS was stuck at 32MB, then you needed some way of letting the user get to all of their hard drive storage. So I'm sure OnTrack made a bundle because there was one of those little 5.25 floppy with every hard drive you bought which installed that device driver on the fly.

Leo: Yup, yup, yup.

Steve: And it did this lie. It told DOS that its sectors were much larger so that its 64K sectors would now span a much larger drive. Anyway, SpinRite, because of its unique position in this, it had to have code to straddle that lie. It had to operate with DOS's view of logical sectors while at the same time working with the actual underlying true 512-byte physical sectors.

So the history of our early PCs is littered, and it really has become littered, with storage size limitations because no one ever expected what happened. The original hard drives were limited to 1024 cylinders, 16 heads, and 63 sectors. So that's 10 bits per cylinder, only 4 bits per head, and 6 bits for the sector. The sector count was 63 because for some reason no one knows, there never was a sector zero. Sectors were always numbered from one. But that set of size limitations imposed a larger 504MB upper limit on hard disk size. If you worked around that limitation, or in order to work around that limitation, early BIOSes used a byte for the head number. So now you could have 255 heads, or 256 actually, zero to 255. That led to all of those wacky cylinder head sector, remember the CHS translation schemes where...

Leo: Oh, yeah.

Steve: ...you were like, oh, my god.

Leo: Oh, what a nightmare that was.

Steve: Yes.

Leo: I can only imagine coding for it. I mean, bad enough we had to use it.

Steve: Exactly. I had to penetrate that translation lie. But even so, the limit then was 7.84GB. And even the early ATA spec which expanded the cylinder count out to 64K cylinders, you know, 65,536, a full 16-bit cylinder number finally, not just 10 bits, it still had only 4 bits in its registers for 16 heads. So more translation needed. You'd translate the heads up and the cylinders down. But it did allow for 255 sectors. So then you could now get 127.8GB using BIOS head translation with an ATA drive. So it's just been, like, one after another short-sighted design which we kludged ourselves around. Oh, and Leo, then came along the monkey wrench of on-the-fly partition compression. Oh, my god.

Leo: Was that Stacker and stuff like that?

Steve: Yes, exactly. Stacker from Stac Corporation. And Double...

Leo: Oh, god. This is all coming - Double Disk, was that it? Double...

Steve: Double something or other. Double Trouble.

Leo: Yeah. Oh.

Steve: Now, because a compressing device driver had been added between DOS and the underlying mass storage, what DOS was seeing was an entirely fictitious drive partition with an amount of free space that was fluctuating based upon the dynamic compression ratio of all the data that had been stored so far. Again, SpinRite needed to bracket any on-the-fly compression driver, sometimes seeing the drive from DOS's bizarre and dynamically changing perspective, and relating that to what was still happening down at the drive's actual true 512 bytes per sector reality. It was all a horrific mess. But SpinRite just had to work in any and all situations. It needed to, and it did, figure all of that out and just deal with it so that users could simply run it no matter what their system was doing, and it would just work.

And one other piece of SpinRite engineering that I recently encountered in the last week and had completely forgotten about was that SpinRite might be writing to a file into its own detailed technical log, recording everything it found and did back onto the same drive it was working on. That drive would have a file system on it, and SpinRite might be low-level reformatting and/or extensively pattern testing the hard drive track containing the sectors of the file system to which SpinRite was writing its log because it's going through DOS, and it has no control over where DOS writes the file. And that could be happening while it was working on the track underneath those file system sectors. And there might be a device driver lying about the size of the sectors, and another one compressing those sectors so that they weren't even really sectors anymore, just elastic storage abstractions.

So what does SpinRite do? I virtualized access to the track that SpinRite was working on. SpinRite would read, and if necessary recover, all the data on a track, lifting it into a RAM-based track buffer. Then I would intercept any DOS and BIOS access which matched the current cylinder and head that I was working on and would redirect those reads and writes to that pseudo track in SpinRite's track buffer. In that way it was possible for SpinRite to write its log file through DOS, through whatever device drivers might be present and in the way, and to give DOS simultaneous access to the same track that SpinRite was currently working on - reformatting, changing its sector interleave, and perhaps pattern testing for defects.

And of course, yeah, the much easier solution would have been to simply tell SpinRite's user that they cannot log to the same drive SpinRite is running on. But since it was possible to reengineer or to engineer a way around that limitation, I had to do it. And SpinRite does. Now, today, of course, all of that is, I don't know, water under the bridge, over the dam or something. The problem is all of that crap is still there, even though none of it is useful and none of it is doing anything any longer. And it's getting in the way because I'm having to constantly take all of that into consideration with everything I do because the hooks, the tests, the double checks for all of those exception conditions are laced throughout the code.

Today, sectors are only 512 bytes. They are never something else. Yet I'm constantly multiplying or dividing by logical sectors per physical sector, or the logical sector size, which are two terms in SpinRite's code everywhere. And of course on-the-fly compression is gone. And everything has become so much simpler, simply by expanding everything's true addressability. Yet all of the code for handling all of the kludges that predate today's relative simplicity remain functional and in the way. And of course there's SpinRite's current limitation. As we know, 32 bits is about 4.3 billion. It's the number of IPv4 addresses on the Internet.

And notice that 32 bits was no longer enough for the Internet, either. Well, it's no longer enough for our mass storage. If we take 4.3 billion, 32 bits, as a sector count, and we multiply it by 512 bytes per sector, that brings us to 2.2TB, the absolute maximum size of any drive whose sectors can be accessed with 32 bits. To get around that, as I already have with the new drivers in the ReadSpeed Benchmark, which SpinRite will be inheriting, all of the rest of SpinRite needs to have its sector addressing expanded from 32, well, I'm going to go to 64, although currently it's 48. The current state of the art in the ATAPI spec for all of our state-of-the-art drives is 48 bits of sector addressing, which up from 32, that's going to hold us for a long time because that's, what, 16 bits more. So it's 64K times 2.2TB. So, yeah, that ought to be enough for a while.

Anyway, for the sake of convenience in code, I'll go to 64, so twice 32. But all of the existing data structures and all the code that operates upon them needs to change. And that's not a problem, but all of the old crap code that serves absolutely no purpose any longer would also need to be changed, if I was going to drag it along, or simply be removed forever. My one goal - and of course the hint is the title of this podcast, "Out With the Old." My one goal is to get SpinRite 6.1 out the door in the shortest possible time. The more I think about it, the more I'm sure that saying goodbye to all of that crap which I once labored over and loved - I mean, it was kind of fun looking at it in the last week - is the way to go.

What this essentially means is that SpinRite 6.1's users are going to wind up receiving a much newer and updated solution, more than just strapping these new drivers onto the existing aging code. And another problem is that SpinRite's traditional one-track-at-a-time approach is deeply embedded throughout the code. I mean, that's, you know, SpinRite was born as a one-track-at-a-time sector re-interleaving tool. And as I've said before, the reason it does data recovery is I realize, if I'm going to low-level reformat a track and move the physical sectors around to different positions in order to improve the system's performance, I only have that one last time to get the data off the track.

So I built the best data recovery that I knew how to build. And of course that ends up being SpinRite's claim to fame still today, even though it's been decades since any low-level formatting was actually possible on hard drives. They're all one-to-one interleaved, and they manage sector defects on their own. So despite that fact, even SpinRite 6 still deals with single tracks at a time. So that has to go out the window also.

As we know, SpinRite's primary deliverables will be that it will once again be able to operate upon drives of any size, meaning 48 bits' worth of sectors - yes, 64K times 2.2TB - and that it will do so with all possible speed. I cannot issue track size operations to drives to get them to run at maximum performance. That's too much per command overhead relative to how fast today's drives are. I need to be transferring the maximum 32,000 sectors at a time. And wait a minute, 32,000. Okay, so that is, I am transferring half of what used to be DOS 3.3's maximum partition size per transfer in the new ReadSpeed Benchmark, and that's what I'll be doing with SpinRite.

And of course all of SpinRite's data recovery has always operated upon exactly one physical sector at a time. But this, too, needs to be rethought in the era of drives which use larger actual physical sectors. Logically they still say sectors are 512 bytes. That has

never changed. But we know that modern drives often actually do have 4096-byte sectors. And SSDs are well known to be storing their data in memory pages which are multiples of many physical sectors. So all of that needs to get fixed.

And of course we then have the future. If this was to be SpinRite's final dying gasp, the goal would be to just make it work and be done with it. But all evidence is that we are at the start, I think, of a significant renaissance for SpinRite. We've learned that today's mass storage engineers have been up to all manner of shenanigans, with the focus of designing products so that only a tolerable percentage will fail before their warranty period has lapsed, and also that any data loss will go unnoticed. This is not to say that these products do not deliver stunning performance and capacity. They do. I feel my age when I think about how large today's drives are and how inexpensive they are.

But the nature of economics has always driven any designs and their designers to squeeze out every last possible drop of performance and capacity, while regarding some failure as inevitable and tolerable, even if that's never what any purchaser feels. Warrantees only refer to the device's functioning, never to the user's data. So my point is all of that old code in SpinRite has no future. And we'll be able to get to a SpinRite 7 after SpinRite 6.1 or 6.2, whatever, if I am able to then take SpinRite's new and stripped down code and move it forward. But I cannot move it forward as is. It's just dragging far too much useless history along.

So the conclusion I've reached is that I should remove all of the creaky old code, which is in the way, during the sector addressing expansion from 32 to 48 bits while dropping the equally creaky old cylinder head sector and track abstractions and incorporate an awareness of true physical sector size into SpinRite's data recovery operations. And in the interest of getting this out the door, I will largely leave SpinRite's funky textual UI as is, which will deliver what everyone most wants, a blazing SpinRite that they can again use on drives of any size, with performance that makes that truly practical.

So anyway, I just - when I opened SpinRite up and began looking at what I was going to have to do, I thought, wow, I really have moved so far away, Leo, from the days of a 32MB partition size limit. And then I was seeing this logical to physical, I go, oh, my god, remember OnTrack that, like, you would use that to set your system up in order to get a partition of a useful size. Wow.

Leo: It's amazing. It really is. The thing is, SpinRite has been around for a long time. Most other programs don't have this kind of history. So you've been dealing with this since hard drives began. What's cool is you're entering the SSD era and being just as useful. I think that's really interesting, myself.

Steve: Yeah, yeah. It's really got me revved up. So I'm going to get SpinRite 6.1 out to everybody, and then on to 7.

Leo: Good. Thank you, Steve. And I love it, frankly, I love it that you're keeping the old text-based interface because why not? Why not? You don't need drop shadows and 3D GUIs or anything. It gets everything done. I actually use a lot of, you know, they call it TUIs, Textual User Interfaces. I use a lot of TUI apps. And I think they're great. Nothing wrong with that. Saves you a lot of time.

Steve: In fact, I was just talking about that over in the SpinRite group. I'm sort of thinking, planning ahead where I'm going to go. And I like the idea of using the Windows PE, the Preexecution Environment, because Microsoft allows it to be used for setup and

recovery purposes. So it falls exactly within my usage. It doesn't have a shell. But, oh, and it will only run for 72 hours, three days. And so that's plenty of time with the new - no, no, it's plenty of time with the new SpinRite.

Leo: Really.

Steve: Because it'll be able to do things in hours.

Leo: Oh, boy.

Steve: Yeah, yeah, yeah.

Leo: Yeah. I didn't realize that it was going to be that much faster. That's fantastic.

Steve: Oh, Leo, it is, yes, it screams.

Leo: Wow. Well, soon. You can go get a copy of SpinRite right now, and a copy of this podcast. Get two things done, kill two birds with one stone, as they say. Go to GRC.com. Pick up SpinRite. When you buy 6.0 now, you're automatically buying an upgrade to 6.1, plus you get to participate in the testing as we get closer and closer to the release. GRC.com. You also get 16Kb versions of this show, 64Kb versions of this show. Transcripts, which is great if you like to read along while you listen. That's all at GRC.com, plus a lot of other stuff, all free, Steve's hangout. You can leave him feedback there at GRC.com/feedback. Or on Twitter he's @SGgrc, and you can DM him there. His DMs are open. So if you've got a question, a comment, a suggestion, a Picture of the Week...

Steve: Or a Picture of the Week.

Leo: A Picture of the Week. Always [indiscernible] your Cheetos, @SGgrc. We've got 64Kb audio plus video of the show at our website, TWiT.tv/sn is the actual direct link. There's a YouTube channel. There's also, well, probably the easiest thing to do is subscribe in your favorite podcast application, and that way you'll get it automatically.

Beginning of the year means it's time for the annual TWiT Survey. We're taking a deep dive into who you are, what you like, what you don't like. It's helpful for us as we design shows, but it's also frankly helpful for us with advertisers. And that's our bread and butter at this point. That's what keeps us going. So if you'd like to help us out, shouldn't take too long. Don't answer any question you feel uncomfortable with. Just go to TWiT.tv/survey21, our 2021 TWiT Survey, TWiT.tv/survey21.

Steve, have a great week. Enjoy your rewatch of "The Expanse." I'll enjoy my first watch. And we'll see you next time.

Steve: Yeah, it's so good. And mostly I'm enjoying being up to my elbows in SpinRite. As you said, it's been around for 40 years. It is venerable.

Leo: When you look at the code, do you, like, remember when you wrote that, or what was going on when you wrote that?

Steve: No.

Leo: No. You don't have - but you do, when you look at it, you look at it and you know what's going on; right?

Steve: Yes. My coding style fortunately was already mature enough. But, I mean, I was leaving comment trails. I had comment blocks. And I've always been a believer in long variable names.

Leo: Nice.

Steve: So everything sort of documents itself. So you won't find my code having a QZD constant or something.

Leo: No, no, no. Do you use a lot of macros? I know MASM supports it.

Steve: Yeah. Yeah, MASM supports macros. And I have a whole bunch. In fact it's one of the things that's made it difficult for me to open source my code because I've got fully custom headers, fully custom macros, my own libraries, and all this stuff that I've built over time.

Leo: Yeah. But that speeds it up. I remember when I was doing assembly coding. If you have good macros, it's almost like high-level coding. You don't have to do moves and jumps as much because you've got a macro that does it all.

Steve: Right.

Leo: Fun. I kind of miss those days. It was really fun. But Steve's on it, you know. That's the key, Steve's on it. We'll talk to you next week, Steve.

Steve: Okay, buddy. Bye.