Security Now! #801 - 01-12-21

Out With The Old

This week on Security Now!

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 is 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, a Sci-Fi note about "The Expanse." And then, inspired by the amazing amount of old code I have rediscovered inside SpinRite, I will take our listeners back to the roaring 80's with a look at how far we have come from DOS v3.3 whose maximum partition size... was thirty-three and a half megabytes.

The Famous "Cheeto" door lock

Username: admin
password: admin

Username: KoLpVXriw
password: I*$j">?ui$5

Browser News

Firefox and Chromium CRITICAL updates

Firefox and all of the Chromium-based browsers, the two most important being Chrome and Edge — but also today pretty much everything else other than Safari — all require updating to remove CRITICAL remote system take over bugs. This would be much bigger news if all of our browsers did not enjoy near real time updating. (And a bit later we're going to see what's underway with those Zyxel security endpoints we discussed last week... which do NOT auto-update.)

https://www.mozilla.org/en-US/security/advisories/mfsa2021-01/

In the case of Firefox, the 2020 CVE 16044 carries the title: "Use-after-free write when handling a malicious COOKIE-ECHO SCTP chunk." The description explains that: "A malicious peer could have modified a COOKIE-ECHO chunk in a SCTP packet in a way that potentially resulted in a use-after-free. We presume that with enough effort it could have been exploited to run arbitrary code."

We don't often talk about SCTP, the Stream Control Transmission Protocol. It's sort of a hybrid of UDP and TCP which was originally intended for use with the telecommunications Signaling System 7 (SS7). Being a hybrid, it DOES run over UDP, while still providing TCP's fault-tolerant, reliable, and in-sequence transport of messages. And like TCP and unlike UDP, it also handled congestion control. And unlike either of them, SCTP provides multi-homing and redundant paths to increase resilience and reliability. It was standardized by the IETF in RFC 4960. And although its first reference implementation appeared in v7 of FreeBSD, support for it is now 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 bug discovery, nor state whether it was actively being exploited in the wild. 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. 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.

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 believe that it's bad enough to rank as CRITICAL, though both Google and Microsoft have classified it as only HIGH SEVERITY. And it's unclear what the backstory is on this one. 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 Google published in October 2020 when it was also classified as only HIGH SEVERITY after its discovery the month before. It's a flaw in Chromium's V8 JavaScript and WebAssembly engine. At the time, the bug was also classified as high-severity. The flaw is identified as an "out of bounds write in V8" originally found in September 2020 by Liu.

Today, neither Microsoft nor Google explain why the October 2020 CVE-2020-15995 bug has popped-up again and in need of fixing. But in any event, an additional 12 Chromium bugs were reported by Google with, not surprisingly, a 1-for-1 duplication echoed by Microsoft. Whoever it is over at Microsoft who job it is to report bugs is Edge has it easy... Copy... Paste. Copy... Paste. Time to go to lunch.

The majority of the bugs were rated HIGH SEVERITY and were predominantly use-after-free bugs and they earned their respective discoverers or teams $20,000 each.

Unlike the SCTP flaw that was fixed in Mozilla, which, due to its nature, requires a watering hole style attack. 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.

To change the page or stay where you are?
7 years ago, Blair Mcbride opened a question about what Firefox's Backspace key should do. He opened a Mozilla bug report (though it’s not a bug, it’s just the common way they manage all issues and any sort). He wrote:

> Pressing backspace does different things depending on where the cursor is. If it's in a text input field, 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 behaviour has been argued For A Very Long Time. It's confusing for many people, but we've assumed it would break muscle memory for many people. However, the muscle memory argument was mostly an assumption - AFAIK, we didn't have useful usage data. Until now.
>
> So: Now we have (or can get) useful usage data to either backup the muscle memory argument, or to squash it once and for all - and remove backspace as a shortcut for navigation.

7 years passed with a long, winding and storied thread accruing comments, observations and opinions until, 12 days ago, the issue was resolved and the long outstanding question for Firefox’s handling of the backspace key was at long last answered:

> We should not ‘overload’ common user actions. 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 of changing the page. This creates a high likelihood scenario for a common user behavior, deleting text, to trigger a less likely user behavior “using a keyboard shortcut to navigate back.” This investment resolves the overloaded function and closes the gap on user behavior.
>
> Firefox is the only browser using the “Backspace” key 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 this shortcut:

- The argument to retain “Backspace” keyboard shortcut for muscle memory for users migrating to Firefox or using Firefox along another browser does not apply anymore
- Edge has now replaced IE as a default Windows browser without a similar shortcut (Alt + Left Arrow is the only keyboard shortcut available for this)
- Chrome also only implements Alt + Left Arrow
- Chrome had this shortcut and removed it because of user data loss risks

The “Backspace” keyboard shortcut on Firefox is by far the keyboard shortcut with highest usage with 40M MAU (monthly active users), well above “Find in page” (16M MAU) or “Page reload” (15M MAU). This raises concerns that our users are suffering useability issues and data loss issues from hitting this keyboard shortcut by mistake.

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 monthy active users who are using Backspace to deliberately go back to the previous page, might be suddenly confused and annoyed to discover that its use for that purpose has finally come to an end.

We're currently at Firefox 84. The change 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 released channel.

The good news is the Boolean setting for it can be turned back on. As usual, put “about:config” into the URL field and hit enter. Then, agree to accept the awesome responsibility that accompanies making any changes behind the curtain and search for the string: browser.backspace — one item will be found. This option was already in my Firefox 84, and it works. Setting it to Boolean '0' which mine was already, gives you page-changing Backspace behavior. Setting it to '1' causes Backspace to be ignored when not entering text. This suggests that release 86 is simply going to be flipping the existing 0 setting to a 1.

So, the first time you hit Backspace and Firefox blows you off, you can return the setting to '0'.

Ransomware

Follow the money...

Security researchers following the flow of bitcoin transactions from the victims of Ryuk's ransomware into the bad guy's pockets estimate that the Ryuk operation has netted at least $150 million. They discovered that Ryuk operators primarily use two legitimate cryptocurrency exchanges to cash out the Bitcoin from paying victims as fiat money.

Two threat intelligence companies, Advanced Intelligence and HYAS, tracked 61 Bitcoin wallets attributed to the Ryuk malware enterprise and discovered that the cryptocurrency moves from an intermediary to the Huobi 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 crypto currency 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 above $5 million or 365 bitcoins. But that payout was not the largest ransom paid. In an earlier report, Advanced Intelligence said that the largest payment confirmed to these attackers was 2,200 BTC, which converted to $34 million at the time. The average ransom value received by the group is 48 bitcoins.

Another highly profitable ransomware gang is REvil (Sodinokibi), who announced through a public-facing representative that they made $100 million in one year from extorting victims. They said that the 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 note that they had ended their "We have all the people we need for now status" and are again seeking to recruit new affiliates into their fold.

Intel: A triumph of marketing over technology

Yesterday, during the 2021 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 Threat Detection Technology (TDT). This Threat Detection Technology, they claim, will be able to <quote> "detect ransomware attacks at the hardware silicon level, many layers below antivirus software." Naturally, that was of interest to me. So I spent some time digging a bit deeper into these claims. And, frankly, it's nonsense. It appears to be pure marketing hype. The press release said:

> "Intel TDT uses a combination of CPU telemetry and ML heuristics to detect attack-behavior. It detects ransomware and other threats that leave a footprint on Intel CPU performance monitoring unit (PMU). The Intel PMU sits beneath applications, the OS, and virtualization layers on the system and delivers a more accurate representation of active threats, system-wide. 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."

What you have there is an example of some creative writing. The biggest malware threat faced by the enterprise, 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 that can finally put an end to this scourge. So, quickly cancel your short position on Intel stock and instead buy as much as you can. Intel's future is suddenly bright again.

Except, of course, none of that's true.

The marketing guys apparently met with the engineering guys and asked: "So, the gen-11 vPro processors have this PMU. What's it good for? How can we sell it?" 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 asked: "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 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:

> "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 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.

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'll double-dip guarantee you that a system that's fully 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 isn't constantly throwing 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 taken or not, memory is being read and written, subroutine calls being made and returned from, and transitions being made among various OS privilege rings. But it's all completely anonymous. There's absolutely no context. Everyone is reading and writing registers, doing math, retrieving and storing to memory and 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 that 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" is uncertain.

The strange case of the Male Chastity Cage
On the lighter side I suppose, unless you're unfortunate enough to be wearing one…
Leo, remember those bizarre IoT chastity devices for men that we briefly covered some time ago? It's the "Cellmate Chastity Cage" ... a bargain at $169 for the small model. As I've taken to saying more recently: "What could possibly go wrong?" Although perhaps "What is wrong with this picture" would be more fitting in this instance...

https://www.qiui.store/product-page/cellmate-chastity-cage

It should come as no surprise to anyone that the software behind these devices 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 judgemental." In an effort to keep this a bit tounge-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".

https://internetofdon.gs/qiui-chastity-cage/

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 3rd-party using mobile app/API.

- Multiple API flaws meant anyone could remotely lock all devices and prevent users from releasing themselves.

- Removal then requires an angle grinder or similar, used in close proximity to delicate and sensitive areas.

- Precise user location data also leaked by API, including personal information and private chats.

- Vendor initially responsive, then missed three remediation deadlines they set themselves over a 6 month period, finally refusing to interact any further, even though majority of issues were resolved in migration to v2 API, yet API v1 inexcusably left 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. It's worth noting that sales of smart adult toys has risen significantly during the recent lockdown. (And I suppose that this brings new meaning to the term "lockdown.")

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 offline 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 that poses even greater potential physical harm (an "internal" chastity device), we have felt compelled to publish these findings at this point.

https://www.pentestpartners.com/security-blog/smart-male-chastity-lock-cock-up/

What they found was that making a request to any API endpoint did not require authentication and that using a six-digit "friend code" would return "a huge amount of information about that user" including their location, phone number, plain text password, and so on. They wrote that: "It would not take an attacker more than a couple of days to exfiltrate the entire user database and then use it for blackmail or phishing."

And, somewhat pre-**dick**-ably, following the disclosure, an attacker started targeting Cellmate mobile app users who controlled the smart toy and locked the chastity device. Victims were asked to pay 0.02 bitcoins, around $270 at the time of the attacks.

And, thus, we witness the rare birth of a new class of ransomware.

Security News

A Solarwinds smoking gun?

Yesterday's post by Kaspersky Labs was titled: "Sunburst backdoor – code overlaps with Kazuar"

https://securelist.com/sunburst-backdoor-kazuar/99981/

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, writing:

In a previous blog, we dissected the method used by Sunburst to communicate with its 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.

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 usage of the FNV-1a hash.

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 & Vo. What the hash can boast of is its very small implementation size and its fully tunable output length. The hash operates in a simple loop: The current hash is multiplied by an FNV_prime value then a byte of input is XORed into the result. That is repeated until all of the input has been consumed.

The Kaspersky report goes into 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.

It finally concludes with:

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 the sleep loop, are still not 100% identical.

Possible explanations for these similarities include:

- Sunburst was developed by the same group as Kazuar.
- The Sunburst developers adopted some ideas or code from Kazuar, without having a direct connection (they used Kazuar as an inspiration point).
- Both groups, DarkHalo/UNC2452 and the group using Kazuar, obtained their malware from the same source.
- Some of the Kazuar developers moved to another team, taking knowledge and tools with them.
- The Sunburst developers introduced these subtle links as a form of false flag, in order to shift blame to another group.

At the moment, 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 is possible that evidence confirming one or several of these points might arise. At the same time, it is also possible that the Sunburst developers were really good at their opsec and didn’t make any mistakes, with this link being an elaborate false flag. In any case, this overlap doesn’t change much for the defenders. Supply chain attacks are some of the most sophisticated types of attacks nowadays and have been successfully used in the past by APT groups.

To limit exposure to supply chain attacks, 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 run third party software.
- Implement regular memory dumping and analysis; checking for malicious code running in a decrypted state using a code similarity solution.

So... Kaspersky is a Russia-based security firm. Does that color their conclusions? Attribution in cyberspace **is** incredibly difficult.

The US Government's position
Last Tuesday the U.S. government's intelligence and law enforcement agencies, including the FBI, CISA, the ODNI, and the NSA jointly published a statement:

"This work indicates that an Advanced Persistent Threat (APT) 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."

Again, the phrase "likely Russian" falls short of definitive. At this point, it seems that the inherent difficulty of attribution in cyberspace rules the day.

SolarWinds News
We also talked about the trouble that the Solarwinds corporation would likely be facing. Predictably, a class action lawsuit has been filed by shareholders of SolarWinds beleaguered stock.

https://www.classaction.org/news/solarwinds-hit-with-securities-class-action-over-statements-in-run-up-to-cyberattack-on-fed.-government

The named defendants in the suit are SolarWindows' 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.

Embedded in the suit was a bit of information that raised my eyebrows. Among the many other grievances enumerated by the plaintiffs was that fact that the SolarWinds' update server used the password: solarwinds123. **Yikes!** That does seem irresponsible... especially for a firm whose product offerings are supposed to be all about security.

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".

https://ks.group/

The front page banner — actually, 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.

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 US 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'll 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 we're going to be seeing more of them.

Zyxel security endpoints under attack

Last Tuesday, right while we were talking about the new threat facing the more than 100,000 Zyxel VPN and other security endpoint customers, GreyNoise Intelligence was observing that the previously unknown "zyfwp" account name had been added to SSH scanners and was being actively probed for across the Internet.

Johannes Ullrich, of the SANS Internet Storm Center (ISC) last week said: "Likely due to the holidays, and maybe because the discoverer of the backdoor account did not initially publish the 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, 45.155.205.86) joined throughout this week.

He said: "The initial IPs scanning for this are all geo-locating 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's infrastructure."

This backdoor account with its now widely public password poses a significant risk because it permits anyone on the Internet to create new VPN accounts to gain access to internal networks or to port forward Internal services to make them remotely accessible and exploitable. This is looking like a horrific disaster.

WhatsApp revises their privacy policy

Up until now, WhatsApp's privacy policy opened with the claim that: "Respect for your privacy is coded into our DNA. Since we started WhatsApp, we've aspired to build our Services with a set of strong privacy principles in mind."

That was then. 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 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, is changing that. One month from now, users will no longer have that choice. Some of the data that WhatsApp collects includes:

- User phone numbers
- Other people's phone numbers stored in address books
- Profile names
- Profile pictures
- Location Information
- Transactions And Payments Data
- Device And Connection Information
- Status messages including when a user was last online
- Diagnostic data collected from app logs

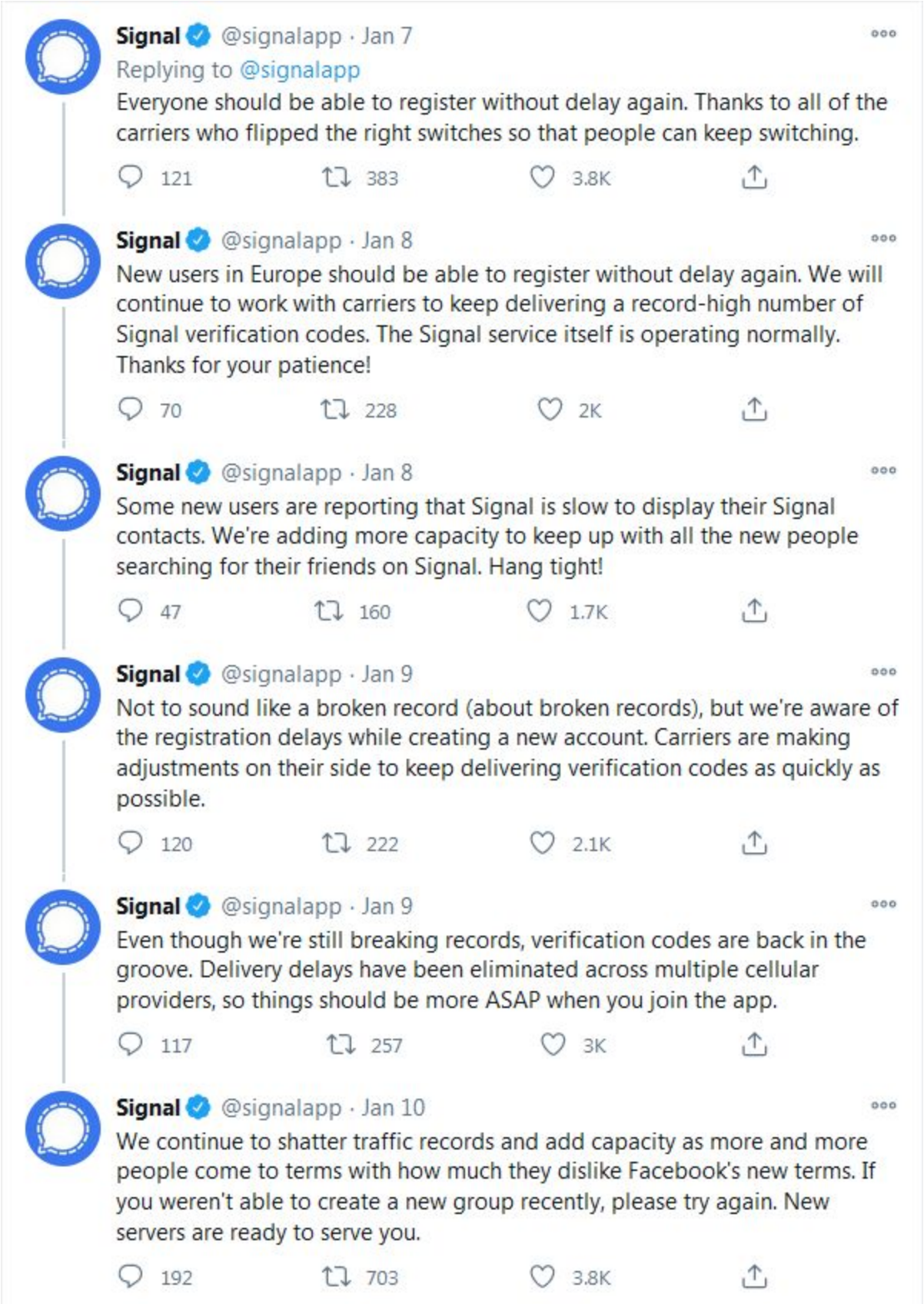
Yeah... they may not see what you're saying, but they have and will shortly be using every other last scrap of metadata. Under the new terms, Facebook reserves the right to share collected data across its family of companies. And in some cases, such as when someone uses WhatsApp to interact with third-party 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 in 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 accounts.

WhatsApp's notification reads: "By tapping AGREE, 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."

Given the supreme lack of subtlety shown, it really does feel like a bait and switch.

Signal reaps the reward
I mentioned that Signal was experiencing a mass influx of ex-WhatApp users. It turns out that Signal's infrastructure was a bit unprepared for the rush.

Bleeping Computer 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. But the surge in new users switching to Signal overwhelmed the verification service, causing significant delays across various mobile providers. That's the sort of problem you like to have...



Sci-Fi

"The Expanse" on Amazon Prime.

Out With The Old

As I've been digging around inside SpinRite, I've been amazed as I've been encountering and remembering how much functionality I managed to cram into this system over time. There were so many kludges perpetrated upon the industry as our early PCs outgrew their modest beginnings. 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 debut, that pace has never let up.

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 had could be made larger. Physical sectors transferred from 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 8-sector multiples rather than the truth. This was fine for DOS, since it really didn't care. But 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 anything) logical vs physical sector truth. And that juggling had to happen for everything SpinRite did.

Remember that the very popular DOS v3.3 had a 32 megabyte limit (actually 33,554,432 bytes) on partition size? Where did that come from? It came from DOS's logical sector number which, back then was 16-bits. 16 bits is 65,536. 65,536 x 512 bytes per sector is 33.55 megabytes. That was the largest possible partition that DOS v3.3 could handle.

And while we're talking about PC history, when I encountered that 33.55 megabyte DOS 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 "Steve's Dream Machine" where every component was carefully selected for cost effectiveness and value. 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 to 64 MB when formatted with an RLL controller (as I recall, the Adaptec 2372). And it turned out that the drive advertised fewer cylinders than were required to deliver a second partition of the maximum possible size under DOS v3.3. Although they weren't advertised, the required additional cylinders were physically there, and they worked. So, one of the coolest hacks I used 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. It was a gimmick, but it was fun and the concept was quite popular back then.

I found the text of my original InfoWorld column online. It was titled: "A Hard Disk Drive for Steve's Dream Machine" http://ivanlef0u.fr/repo/madchat/esprit/textes/old_txt/tech/disks2.txt

Excerpting the relevant portion:

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 33.4 megabyte DOS partitions! They couldn't be any bigger! Sixty-seven fast megabytes in an inexpensive half-height drive is hard to beat!

So, 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. I think there was a company called “OnTrack” that bundled its driver which did that with many hard drives. But in any event, SpinRite had to have code to straddle that lie, operating with DOS’s view of logical sectors while at the same time working with the actual underlying 512-byte physical sectors.

So the history of PCs is littered — and it really has become litter — with storage size limitations because no one expected what happened. The original hard drives were limited to 1024 cylinders, 16 heads and 63 sectors. 10-bits for cylinder, only 4 bits for head, and 6 bits for sector. The sector count was 63 because for some reason no one knows, there never was a sector 0. Sectors were numbered from 1. But this imposed a 504MB upper limit on HDD size. If you worked around that limitation, early BIOSes used a byte for the head number, so you could now have 255 heads. That led to all of those wacky cylinder/head/sector (CHS) translation schemes. But even so, the limit then was 7.84 GB. And even the early ATA spec, which expanded the cylinder count to 65,536, still only had 4 bits in its registers for 16 heads. More translation needed. But it did allow for 255 sectors, so you could get to 127.8 GB using BIOS head translation with an ATA drive.

And then came along the monkey wrench of on-the-fly partition compression. Oh my god. 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 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 it all out and deal with it.

And another piece of SpinRite's engineering that I recently encountered and had completely forgotten about, was that SpinRite might be writing to its 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 while it was working on the track underneath those 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 really even sectors anymore -- just elastic storage. So what does SpinRite do? I virtualized access to the track that SpinRite was working on. SpinRite would read and, if necessary recover, and data on a track, placing it into a track buffer. Then. it would intercept any DOS and BIOS access which matched the current cylinder and head, and would redirect those reads or writes to that track to its track buffer. In that way, it was possible for SpinRite to write its log file through DOS, through whatever drivers might be present in the way, and to give DOS simultaneous access to the track that SpinRite was currently working on -- reformatting, changing its sector interleave, and perhaps pattern testing for defects.

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 engineer a way around that limitation, what choice did I have?

Today, of course, all of that is water under the bridge, or over the damn, 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 is getting in the way because I'm having to constantly take all of that into consideration with everything I do, because the hooks, tests and doublechecks for all of those exception conditions are laced throughout the code. Sectors are ONLY 512 bytes, never something else. Yet I'm constantly multiplying or dividing by "LogicalSectorsPerPhysical" or “LogicalSectorSize”. 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 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. Notice that 32 bits was no longer enough for the Internet, either. If we take 4.3 billion as a sector count, we multiply it by 512 bytes per sector, which brings us to 2.2 terabytes... the 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 to 64 bits. So all of the existing data structures - and the code that operates upon them - needs to change. That's no problem, but all of the OLD CRAP CODE that serves absolutely no purpose any longer also either needs to be changed... or simply removed forever.

My ONE GOAL is to get SpinRite v6.1 out the door in the shortest possible time. The more I think about it the more sure I am that saying goodbye to all of that crap — which once labored over and loved — is the way to go. What this essentially means is that SpinRite’s users are going to wind up receiving a much newer and updated solution, more than just strapping these new drivers onto the aging core.

Another problem is that SpinRite's traditional one-track-at-a-time approach is DEEPLY embedded throughout the code. SpinRite's cylinder, head, and sector, current operation location display has been the purest of abstraction since SpinRite 6's launch back in 2004. So that hasn't been any problem until now, since even though it has long been abstract, SpinRite has continued to operate upon one "Logical Track" at a time.

But, as we know, SpinRite's primary deliverables will be that it will once again be able to to operate upon drives of **any** size, and that it will do so with all possible speed. I cannot issue "track size" operations to drives to obtain maximum performance. I need to be transferring

16 MB (32K sectors) at a time... that’s what the ReadSpeed benchmark and its drivers do.

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 much larger actual physical sectors. And SSDs are well known to be storing their data in memory pages which are multiples of many physical sectors.

And then we 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 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. But the nature of economics has always driven any designs 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. Warranties 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 will be able to get to a SpinRite 7 much faster 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 dragging far too much 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 move from 32 to 48 bits, while dropping the equally creaky old cylinder/head/sector & track abstractions and incorporating an awareness of true physical sector size into SpinRite's data recovery operations.

In the interest of getting this out the door, I’ll 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.