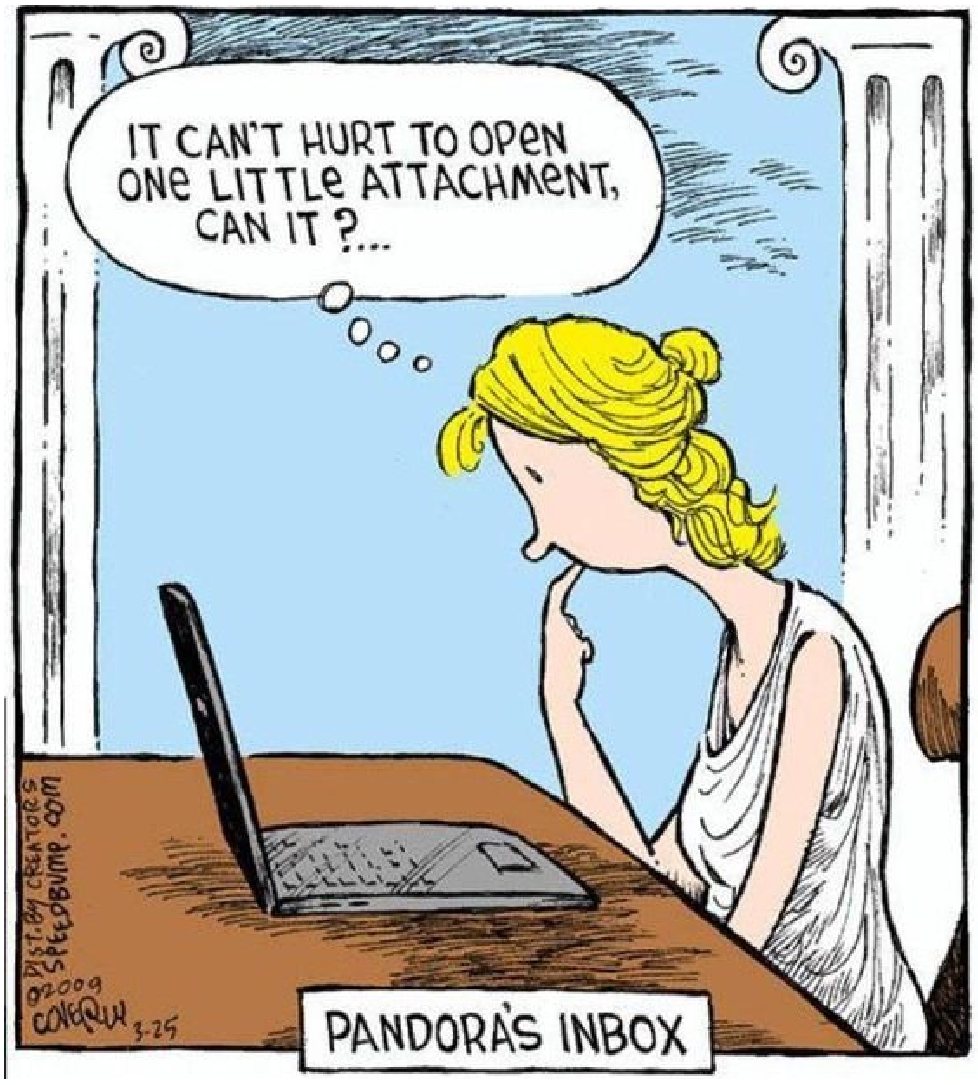
Security Now! #834 - 08-31-21

Life: Hanging by a PIN

This week on Security Now!

This week we'll start out by clarifying the terms credit freeze and credit lock. Then we have news of the T-Mobile breach from its perpetrator. We examine the evolving and infuriating question of where will Windows 11 run and we look at yet another newly revealed attack against Microsoft's Exchange server known as ProxyToken. I wanted to clarify a bit about Tailscale's source openness, and touch on the disturbing revelations shaking the mass storage industry with SSD performance being deliberately reduced once they've been well reviewed and adopted. I'll update our patient SpinRite owners on my recent work and progress, we'll touch on some cellular phone terminology, then conclude by considering the power of the PIN and look at just how much damage it can do.


IT CAN'T HURT TO OPEN ONE LITTLE ATTACHMENT, CAN IT?...
PANDORA'S INBOX
©2009 DIST. BY CREATORS SPEEDBUMP.COM
COVERLY 3-25

Credit Freeze vs Credit Lock

I want to begin this week by first and immediately correcting my mistake last week about the terms “Credit Freeze” versus “Credit Lock”. It turns out that the terms matter and I got them somewhat mixed up. It's a credit **FREEZE** that most people will want to use, **not a credit lock**.

A “Freeze” is the term used for the federally mandated, no-cost option which **all four** major bureaus must provide to prevent queries made from would-be creditors from being honored.

By comparison, the term credit “Lock” is used to designate an optional **product** which may take the form of a costly subscription and which may be made available by the various credit reporting agencies with differing features from one company to the next. It's the Lock product, for example, that has the additional bells and whistles and which might be tied to mobile phone apps to allow rapid locking and unlocking. I made the mistake of not digging deeply enough into the terminology before talking about it last week. So thank you to all of our listeners who DID know better and who let me know that I'd gotten things somewhat muddled.

So the FREEZE is what everyone should enable. And being a federally mandated service it costs nothing. It also appears that the temporary auto-relock is part of the Freeze service, making it possible to briefly lower one's shields. I also noted some options to temporarily and selectively allow queries to be honored for specific and specified would-be creditors. If this begins to sound a bit like you're programming a firewall, that analogy is pretty accurate. In firewall parlance, rather than the “Permit Any” rule we want to run with the “Deny All” and then optionally add rules to permit specific creditors to query for and receive reports.

And you may have noted that I mentioned “the four” major credit reporting bureaus. A fourth upstart bureau called “Innovis” has been added to the traditional big three: Experian, Equifax and TransUnion. So you'll also want to place a reporting FREEZE on your records with Innovis as well since someone having access to all of your personal data from a data breach might deliberately choose to attempt to obtain credit from some lender who is know to use Innovis specifically because, being lesser known, they are less likely to have had their reporting frozen.

And speaking of the T-Mobile breach...

Security News

T-Mobile

Thanks to the fact that the attacker, a US citizen, believes that he's currently outside the long arm of U.S. Law enforcement, we're learning quite a lot about the who, what and why of his quite successful data exfiltration attack on T-Mobile. And none of what we're learning flatters T-Mobile's cybersecurity.

The Wall Street Journal has been chatting with the purported attacker via Telegram for a while. They've confirmed that his name is John Binns. John is a 21-year-old US citizen of Turkish descent who relocated from the US to Turkey three years ago. John was reportedly discussing details of the breach before they were widely known and T-Mobile received their first indications of trouble when they were notified of the breach by Unit221B LLC, a cybersecurity company that monitors the dark web for their own purposes.

John told the WSJ that his attack against T-Mobile was conducted from the comfort of his home in Izmir, Turkey, where he lives with his mother of Turkish descent. His American father died when he was 2, and he and his mom moved back to Turkey three years ago when John was 18. He reportedly uses the online handles IRDev and v0rtex, among others and he’s alleged to have an online track record that includes some participation in the creation of a massive botnet that was used for online DDoS attacks four years ago when he was still in the U.S and 17 years old.

John told the WSJ that he penetrated T-Mobile’s defenses last month, in July, after scanning the company’s known internet addresses, looking for weak spots and using what the Journal referred to as “a simple tool available to the public.” During his perusal of T-Mobile's public-facing IP space last month, John discovered an exposed and unprotected router. This router reportedly had a different configuration from T-Mobile's other public routers, though details are still scarce. But from that router John said that he had managed to break into T-Mobile’s large data center located outside East Wenatchee [wuh·na·chee], Washington. Inside the Wenatchee facility, John said he had access to more than 100 servers containing the personal data of millions. By August 4th he had exfiltrated millions of files thanks to what he told the Journal was the mobile phone seller’s pathetic security. Note, also, that the company that's big on the color magenta, but apparently not so big on cybersecurity, never had any idea that he was loose inside their network, roaming around freely at will.

Observers within the cybersecurity community noted that the fact that the theft included records from prospective clients as well as former, long-gone customers, demonstrates the extreme degree to which no one inside T-Mobile was practicing any data management hygiene.

And I’ll add to that, that the data John was able to make off with was not encrypted. That sort of sensitive data at rest should be encrypted so that only the system that’s validly retrieving it for its proper business purpose should be able to decrypt and read it. The files themselves should never be stored in simple exportable and readable plaintext. But T-Mobile’s was.

When you consider a publicly exposed router that can be broken into, apparently no network intrusion monitoring and response—because they never knew—and a complete lack of long term data management and policy, the fact that this is the 6th such data breach in just the past few years should not be surprising.

As we've discussed before, I've always taken the position that the person in charge should not necessarily be fired when something bad happens on their watch because those can be teachable moments and the result might be much improved security in the future. But now I’m not so sure. T-Mobile is making me rethink the beneficent tolerance approach.

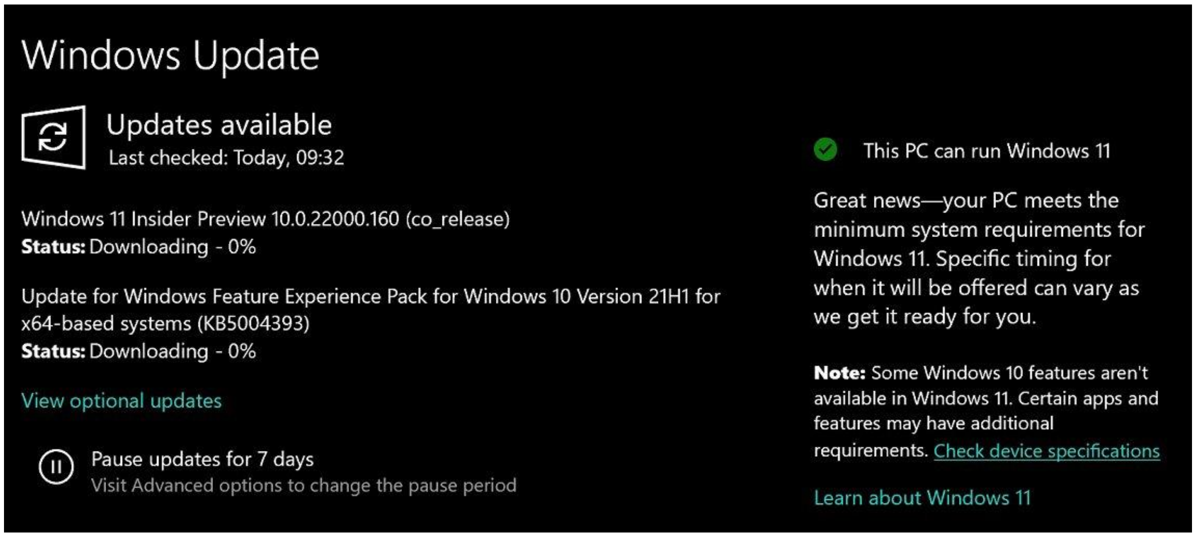
Where will Windows 11 run?
At the moment, no one seems to be sure... and Microsoft isn't helping. The best advice I have is to take the time to check-in with the maker of your machine, motherboard or whatever, to see whether they're offering any updates to increase compatibility with Windows 11. We noted two weeks ago that Asus had deliberately updated the firmware of 207 of their motherboards specifically to ease their users' move to Windows 11. So it's worth checking to see whether whatever system you're using might have some similar updates available.

Microsoft continues to confuse. They appear to be holding onto their baseless “must have TPMv2.0” requirement, even though Windows 10 has no such need. And we hear that they’re gradually adding older Intel chips to the approved list. But Intel chips are famously backwards compatible. So I'm at a loss to understand what's really going on there, since no one imagines that Windows 11 is really so different from Windows 10. They added the Intel Core X-series, Xeon W-series, and the Intel Core 7820HQ chips. However, Microsoft also stated that no AMD Zen 1 processors would be compatible... so there!

When you look back at the history of Microsoft's major version upgrades for Windows, the jumps from NT to 2000, 2000 to XP, XP to Win7, Win7 to Win8, and Win8 to Win10 have all brought actual substantive changes. This current move from 10 to 11 doesn't have that feeling at all. So it's difficult to understand why Windows 11 cannot simply run everywhere Win10 does…

... unless Microsoft, for some reason, just doesn't want Win11 to run everywhere. But that will create exactly the sort of stratification that Microsoft worked so hard to avoid when they attempted to force everyone to move to Windows 10 whether they wanted to or not.

Apparently Microsoft will be attempting to resolve some of this confusion by adding a Windows 11 compatible message to the Windows Update screen. I have a snapshot of one in the show notes:


Windows Update

Updates available
Last checked: Today, 09:32

Windows 11 Insider Preview 10.0.22000.160 (co_release)
Status: Downloading - 0%

Update for Windows Feature Experience Pack for Windows 10 Version 21H1 for x64-based systems (KB5004393)
Status: Downloading - 0%

View optional updates

Pause updates for 7 days
Visit Advanced options to change the pause period

This PC can run Windows 11

Great news—your PC meets the minimum system requirements for Windows 11. Specific timing for when it will be offered can vary as we get it ready for you.

Note: Some Windows 10 features aren't available in Windows 11. Certain apps and features may have additional requirements. Check device specifications

Learn about Windows 11

“Great news—your PC meets the minimum system requirements for Windows 11. Specific timing for when it will be offered can vary as we get it ready for you.”

Okay. But *“Specific timing for when it will be offered can vary as we get it ready for you.”*

What the hell does that mean?
“We’re not really sure when Windows 11 will be ready to run on your machine, despite the fact that we’re announcing here the good news that your machine will be able to run it... just not when.”

Somehow, Microsoft appears to have succeeded at completely removing **all** of the science from computer science. Now it’s…

“Well, no one around here who we’ve been asking seems to be completely certain about when exactly this next big release of Windows 11 may be ready for your particular machine. Because we haven’t really figured out what we’re going to do yet.”

Yeah, wake me up after you have figured this out. I can’t wait! What a cluster you-know-what.

Oh, and just now, when I booted the Win10 machine that I only use to connect to TWiT for this podcast, none of its desktop icons appeared, and the on-screen clock that auto-starts wasn’t there. But this is Win10 right?; where all of the science has been removed? So I just shrugged and restarted and this time Windows felt more like finishing up, so I got all of my desktop icons back and everything appears to be present and accounted for. I can’t wait to see what Windows 11 has in store for us.

ProxyToken

So, first we had ProxyLogon, the original mess with Microsoft's Exchange server that they didn't patch until its use in active attacks became public. Then last week we discussed ProxyLogon's kissing cousin, ProxyShell. Similar but different. And today we have what's being called Exchange Server's ProxyToken vulnerability.

It was finally patched — after more than three months — with July's patch Tuesday last month. Microsoft was originally informed of this vulnerability, through the Zero-Day Initiative, by a Vietnamese security researcher on April 5th of this year. The ProxyToken vulnerability would, for example, allow an attacker to surreptitiously add an email forwarding rule to a user’s mailbox so that all emails addressed to the victim will also be sent to an account controlled by the attacker.

The vulnerability essentially allows a remote attacker to bypass authentication and make changes to an Exchange email server’s backend configuration. The flaw was reported, as I said, through the Zero-Day Initiative program and it exists due to a pair of flaws in the Exchange code:

- Requests that contain a non-empty cookie named “SecurityToken” that are redirected from the frontend to the backend are not authenticated.
- HTTP 500 error responses expose an Exchange control panel canary token.

By combining these two oversights, its discoverer explained that a ProxyToken attack is possible and that attackers can easily make requests to any part of the Exchange backend, including its users’ control panels and settings. And since the details of this attack went live yesterday on the Zero-Day Initiative blog, server owners should expect threat actors to weaponize this vector.

https://www.zerodayinitiative.com/blog/2021/8/30/proxytoken-an-authentication-bypass-in-microsoft-exchange-server

Weaponizing the vector is exactly what happened last month when attacks against Exchange servers took off after details about the ProxyShell vulnerability were published online. Those attacks quickly escalated in a matter of days and today, as we noted last week, a new ransomware operation known as LockFile is abusing Exchange servers to encrypt corporate networks.

Errata

Tailscale Open Source?

I received a DM tweet from a listener who wrote: *“Hi Steve! Maybe I missed it, but Tailscale is open source”* and he provided a link to Tailscale on Github: https://github.com/tailscale/tailscale

And it's true that much, though not all, of TailScale is open source. Tailscale explains: *“This repository contains all the open source Tailscale client code and the tailscaled daemon and tailscale CLI tool. The tailscaled daemon runs primarily on Linux; it also works to varying degrees on FreeBSD, OpenBSD, Darwin, and Windows.”*

They provide a link to their Tailscale for Android client: https://github.com/tailscale/tailscale-android and for non-Android Tailscale clients they write: *“The macOS, iOS, and Windows clients use the code in this repository but additionally include small GUI wrappers that are not open source.”*

So, just to be clear, I have not yet dug into any of this. But it does look as though someone who wanted to take more responsibility for setting things up with Tailscale, and who wanted to roll their own solution could definitely do that using the open source code provided within Tailscale’s repository. Though, presumably, such users would also then be responsible for keeping those things up to date.

Tailscale provides links to both their stable and unstable package builds. They provide support for a huge array of Linuxes and also for a Windows installer: https://pkgs.tailscale.com/stable/

So I wanted to give the Tailscale folks credit for and to acknowledge the open source aspects of their offerings.

Miscellany

SSD Bait & Switch

Both Tom's Hardware and ExtremeTech have been following the growing controversy over the practice that's been discovered among an increasing number of SSD makers who have been caught initially releasing a new high-quality product for review and analysis by the tech publications and presumably by their large OEMs for inclusion in future systems — and then once that's been done, quietly replacing their initial fast and high-quality semiconductors with significantly lower cost and lower performance components while not changing the device's part number in any way.

What this means for us is that the important and typically carefully considered opinions of the reviewers of these products may not be reflective of the devices that we eventually purchase after relying upon such reviews. And it also means that tremendous commercial pressure is placed upon those (unfortunately fewer and fewer) companies who are resisting this fraudulent bait & switch behavior.

Though this is off topic for this podcast, I obviously have a huge personal interest in the whole topic of mass storage and its performance, reliability and recoverability. And I know that our many tech savvy listeners will too.

A few weeks ago on August 16th, ExtremeTech's Joel Hruska posted a piece titled: **Buyer Beware: Crucial Swaps P2 SSD’s TLC NAND for Slower Chips**
https://www.extremetech.com/computing/325824-buyer-beware-crucial-swaps-p2-ssds-tlc-nand-for-slower-inferior-qlc-chips

Crucial has come under fire after a retest of its well-reviewed P2 SSD demonstrated that the company has swapped from its launch design to a much inferior product. This is not the first time SSD manufacturers have been caught bait-and-switching customers in this fashion and it’s deeply frustrating to see companies willing to subvert their own review process.

The scheme goes like this: Sample an SSD out to reviewers and spec it reasonably well. Once all the reviews are in, swap out the components for inferior products that are not as power-efficient and/or do not offer the same performance. That’s what Tom’s Hardware found when it investigated Crucial’s P2 NVMe M.2 SSD after reviewing the initial part shipped by Crucial.

Crucial has swapped the TLC NAND it originally shipped with QLC NAND — and not terribly good QLC NAND, at that. The new version of the P2 has two fewer NAND chip packages than the original, and significantly fewer total dies. This reduces the total potential bandwidth the SSD controller can achieve and further harms the performance of the 500GB drive. Average power consumption on the QLC drive is lower, at 1.49W, but total power efficiency is actually worse because the savings do not make up for the dramatically slower performance. If full drive performance on the P2 was already bad, it’s downright abysmal on the P2 with QLC NAND.

So that was on the 16th. Exactly one week ago, on August 24th, Joel followed that up with another piece titled: **Western Digital Caught Bait & Switching Customers With Slow SSDs**
https://www.extremetech.com/computing/326200-western-digital-caught-bait-and-switching-customers-with-slow-ssds

When I wrote about Crucial’s decision to swap inferior NAND flash into its products without updating the reviewer community or announcing a separate SKU, I hoped the problem was a one-off. While this has happened before, it’s typically been the exception, not the norm.

Guess that was too much to hope for. According to a report from Chinese tech site Expreview, the WD SN550 Blue — which is currently one of the best-reviewed budget SSDs on the market — has undergone a NAND lobotomy. While the new SSD variant performs on-par with the old drive that WD actually sampled for review, once you exhaust the SLC NAND cache, performance craters from 610MB/s (as measured by THG) to 390MB/s (as measured by Expreview). The new drive offers just 64 percent of the performance of the old drive.

This is unacceptable. It is unethical for any company to sample and launch a product to strong reviews only to turn around and sell an inferior version of that hardware at a later date without changing the product SKU or telling customers that they’re buying garbage. I do not use the *term "garbage" lightly, but let me be clear: If you silently change the hardware components you use in a way that makes your product lose performance, and you do not disclose that information prominently to the customer (ideally through a separate SKU), you are selling garbage. There's nothing wrong with selling a slower SSD at a good price, and there's nothing right about abusing the goodwill of reviewers and enthusiasts to kick bad hardware out the door.*

And sadly, Joel followed this us with his latest in this series last Friday the 27th by posting: **Samsung Is the Latest SSD Manufacturer Caught Cheating Its Customers**
https://www.extremetech.com/computing/326377-samsung-is-the-latest-ssd-manufacturer-cheating-its-customers

In the past 11 days, both Crucial and Western Digital have been caught swapping the TLC NAND used for certain products with inferior QLC NAND without updating product SKUs or informing reviewers that this change was happening. Shipping one product to reviewers and a different product to consumers is unacceptable and we recently recommended that readers buy SSDs from Samsung or Intel in lieu of WD or Crucial.

As of today, we have to take Samsung off that list. One difference in this situation is that Samsung isn't swapping TLC for QLC — it's swapping the drive controller + TLC for a different, inferior drive controller and different TLC. The net effect is still a steep performance decline in certain tests. We've asked Intel to specifically confirm it does not engage in this kind of consumer-hostile behavior and will report back if it does.

Joel's post goes on to show photos of the peeled off top label off of a Samsung 970 EVO Plus SSD to reveal very different chips underneath. Of course, there's no problem with them doing that. They're free to put whatever chips they like on their products. But if that's done after the drives have been reviewed to shave their cost and the user's performance, I agree with Joel that's not okay.

And just to put a bow on this, Western Digital said: "In June 2021, we replaced the NAND in the WD Blue SN550 NVMe SSD and updated the firmware." thus confirming that an undocumented parts change they made was responsible for a 50% reduction in writing performance. They said: "At the time, we updated the product data sheet. For greater transparency going forward, if we make a change to an existing internal SSD, we commit to introducing a new model number whenever any related published specifications are impacted."

So, I wanted to put this on everyone's radar to make sure that our listeners knew that this was apparently going on within the industry. Everyone knows that I believe in benchmarks and in performance testing. GRC's DNS Benchmark has become the industry standard tool with more than 7 million downloads. And the first thing I did with SpinRite's new driver technology was to create the ReadSpeed drive benchmark. One of the things we immediately learned was that there were some very weird things going on inside our SSDs. They do not behave at all like the solid state RAM we would wish they were. Now is not the time for me to dig into those particular weeds. But everyone can rest assured that this has my attention and that a future SpinRite is going to be quite revealing.

SpinRite

And speaking of SpinRite: After an unusually long code writing stint — without doing any testing — I recently switched back to testing and debugging all of that new code I've been writing.

Since writing in assembler allows me to reassemble and link my code in about half a second, the cost to rebuild my entire project is minimal. In fact, I use that to check for typos on the fly. So the rhythm that's developed for me over the years is "fast iteration," where I'll write a chunk of code then stop to immediately test it to verify that it does what I need. Then I'll move forward knowing that what I've left behind is at least mostly ready for the world. This creates a solid foundation for whatever follows. But I had stopped working that way when I began the rework of SpinRite's device driver model to "abstract" all of SpinRite's drive interaction behind a single custom IO function. I stopped iterating because for a long time I haven't been able to. I needed to pretty much take SpinRite down for the rework and hold my breath while I reassembled it in a very different way. Then once it was back up I thought that I should just keep pushing forward rewriting the code that would use the new IO abstraction system. I did enough of that to see that my first design needed a bit of tweaking. That regarded SpinRite's handling of the drive's built-in error correction which I realized should have been transparent at the abstraction layer since there's no reason to return an unfinished request to our caller if we've been able to autonomously correct and obtain the data. So I reworked the five different drive-interfaces to autonomously handle their drives' reports of corrected errors.

But what wasn't sitting well was that I had written so much code that hadn't yet had any testing. So last week I decided to stop writing and to switch back to testing and debugging. I've been having a wonderful time verifying expected behavior, when everything works as I designed, and tracking down the causes for unexpected mysteries when things don't work as I expect. I posted over in GRC's SpinRite.dev newsgroup last week, that I could easily be pulling all-nighters because chasing down these little mysteries was so much fun and was so compelling. But I noted that I was no longer coding as a teenager and that I had learned that if I stayed up all night, I'd pretty much be useless the next day, and thus the extra time that I'd borrowed from the night would wind up returned immediately, resulting in zero net gain. So I'm forcing myself to quit at the end of each night, knowing that everything will still be there waiting for me when both I and my coffee are fresh in the morning.

Overall, it's going really well and although I didn't foresee the path I'd be taking when I began, I'm very pleased with all of the decisions I've made to get to this point. Although the outside of SpinRite **will** reveal a few signs of an entirely new SpinRite underneath, it will largely look the same. It will be blazingly fast and will work on drives of any possible sector count. And thanks to its new IO abstraction model, it will also be ready to graciously accept the native USB and NVMe support that will be added after the move off of DOS so we're able to boot on newer systems that have removed support for the BIOS and only boot UEFI-based operating system code.

So let's take our last break and then I want to point out how, and the degree to which, our lives really are hanging by a PIN...

Life: Hanging by a PIN

The subtopic for today's podcast would be:

Our cellular phones as a CRITICAL weakest link in the chain.

In addition to the best general purpose advice I give to everyone—which is to operate your life with your credit reporting FROZEN at all four of the credit reporting bureaus, after this breach, T-Mobile subscribers should be sure to change their account's PIN, and when doing so, to make their new PIN as long and complex as possible within the limitations of a PIN and to use digits having the highest possible entropy that have nothing to do with their life. Never use any date of significance, or anything that someone who knows you might guess, since a determined attacker might have been able to gather sufficient information to effectively know you.

Since the PIN was one of the items of information stolen that's under a subscriber's control, and since it's CRUCIAL for verifying your identity to your cellular carrier, it really **MUST** be changed.

All of the various cellular carriers now offer some form of "SIMjacking" protection. This is also known as "SIM swapping" or "port-out scamming." It occurs when a scammer contacts your cellular provider and pretends to be you. The trouble is, in the case of T-Mobile, a scammer will be armed with everything T-Mobile knows about you—including the account PIN that was stolen.

I recently set up a new phone with my provider, Verizon. The battery of my beloved iPhone 10, which I'd owned for years, had outgassed and ballooned, popping the screen out and causing it to bend alarmingly. This was actually the second time that had happened, so I had Apple replace the battery to create a hand-me-down, having decided that it was time to move to an iPhone 12.

So that I wouldn't be without a phone during the repair, I first purchased the new 12 and had Apple deliver it using their incredible "By the way, that's us knocking at your front door with your new phone" service. It literally took all of an hour to have the new phone delivered. So that process was painless. But today, thinking back over just how painless the process was, in the context of the T-Mobile breach, I realize that it was a terrifyingly simple thing for me to move my cellular service over to the new phone.

The only thing Verizon needed from me, that wasn't otherwise generally available public knowledge, was my account PIN which, yes, was also part of the T-Mobile data breach.

That PIN was effectively my entire proof of identity. They didn't need eMail, nor for me to first respond through the previous phone (which I had already decommissioned) and if the phone was claimed to be dead, lost or stolen they still need to be able to move forward. So everything boils down to your account PIN. I simply provided the few digits of my PIN, which they confirmed matched the one they had on file, then I read off long strings of numbers (my new phone's ICCID and IMEI) and just like that, my new phone was live — with my phone number.

Once a bad guy has taken over your phone number, your actual phone will lose service. That's your first clue that life is about to become much more complicated... and not in a good way.

Naturally, any of your online services that still rely upon SMS text messages sent to your registered phone number—or can somehow be made to rely upon sending a text message—will immediately be subject to compromise and takeover. Your eMail provider cannot send you an account recovery eMail if you claim to have forgotten your eMail account password. So they'll have your phone number on file in order to send a text message to your phone for emergency account recovery. And we know that Google, for example, is pretty good about prompting you from time to time to make sure your phone number with them is still current and correct. Guess why?

But now, the attacker who used your PIN to impersonate you to your cell provider to get your phone number moved over to his phone, will receive that emergency account recovery text after claiming to have forgotten your eMail logon password. And the first thing they'll do will be to change your eMail password in case the account recovery didn't automatically change it for them. Now you're locked out of your own eMail.

So next, they'll start accessing your other various services, clicking the "Oh my, I forgot my password!" link and thus obtaining access to any of your accounts when that account recovery link is sent to **your** eMail, which they now control.

In fact, if your password manager provides eMail or SMS-based account recovery, your attacker can go right to the source, obtaining your entire master password archive to learn not only every account name and password, but also everywhere you have accounts... and your password manager will log them right on.

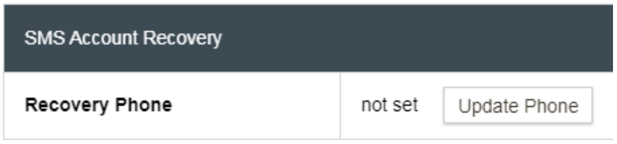
Or if you've decided to simply have your browser memorize your logons, they can now logon as you on the same make and model browser, have that browser synchronize all of its settings, history... and passwords, then browse though its long list of saved usernames and passwords for everywhere you have accounts.

It really is a nightmare scenario. And this entire cascade of events is only prevented by someone with ill intent **not** knowing your incredibly weak, short, decimal 4-digit PIN — which probably IS currently your birthday or year of birth or anniversary or street address number. Because, after all, you wouldn't want to forget it! We're dealing with cellular phones that are also being used by those who have no regard for security. So the security of the entire system has been reduced to serve the lowest common denominator user. Unfortunately, because an SMS-enabled cellular phone is also the one thing that everyone now has, this weakest link has also evolved to become our universal identity verifier — and it deserves no such respect.

I went over to Verizon's FAQ page about PINS. There, Q&A question #1 is: *"What's an Account PIN and why do I need one?"* and they provide the answer: *"Having a PIN helps to keep your Verizon mobile account and personal information secure.* ***It's the primary way we verify you*** *as the Account Owner when you contact Customer Service."* and it goes on to explain: *"If you don't have an Account PIN when you contact Customer Service for account changes or information, you'll be asked to create one to continue."*

So take a moment to think about this single-point-of-failure vulnerability, and the cascade of disaster that flows from someone who's somehow able to obtain access to your phone number.

I just changed my PIN at Verizon—I really just did—because this has all made me realize that I haven't been taking the lack of security of my cellular phone account PIN seriously enough. And I'm using LastPass. I also just removed my phone from LastPass' SMS account recovery where I **did** have it configured because *"Hey! Extra security backup! Right? **Wrong!**"*


SMS Account Recovery

Recovery Phone — not set

There are times when convenience can create convenience for the wrong person. I'll gladly take responsibility for **never** forgetting my master LastPass password. Actually, I can't forget it because I've never known it. It's a bizarre string of nonsense that is carefully written down and stored offline. And that highlights my point: The weakest link in the chain protecting my master password vault is **not** my insanely long password. It's the fact that I had deliberately established a weak SMS-based backdoor into my account protected by a 4 decimal digit PIN and interactions with a very non-native English speaker, who claims her name is Nancy.

So yeah, if you're a T-Mobile subscriber, oh my god, change your PIN immediately, if not sooner.

And if by some chance you never bothered to assign a PIN to your account, thinking "ehhhh…" then there truly is nothing protecting your cellular account — and then your entire life — from hostile takeover.

And given the general critical weakness of the security of our cellular phone accounts, it might be worth taking a moment to seriously reconsider—as I just have—the value of enabling SMS account recovery for your most critical authentications. I really did just disable that insecure recovery feature for my password manager and I feel a bit of relief. Talk about reducing one's attack surface.

I mentioned earlier that all of the various cellular providers offer some form of SIM jacking protection. For all of the reasons I've just described I'm unconvinced that they are worth the trouble, since cellular carriers still need to be able to respond to: *"I lost my phone, forgot my PIN, and my phone is the way I get eMail."* But even so, nothing would be lost by enabling your provider's anti-SIMjacking feature on your account. If nothing else, it might give "Nancy" who's in the process of helping the attacker to ruin your life a bit of pause before taking the attacker's unsubstantiated word for who they are.

T-Mobile calls their solution "Account Takeover Protection." I have a link in the show notes, or you can just Google "T-Mobile account takeover protection" to find it:
https://www.t-mobile.com/support/plans-features/account-takeover-protection

AT&T has this under their "Manage extra security" option, then look for the "Wireless passcode" section. And if your carrier is Verizon, you dial *611 and ask to place a **Port Freeze** on your account. Again, it's unclear what true security and protection may be afforded, but anything that purports to keep your phone number associated with its current handset seems worth turning on. The consequences of it being maliciously moved, as we've just seen, can be too devastating.

ICCID, IMEI and IMSI — Oh My!!

While we're on the topic of cellular phones, a review of those wacky long strings of identifier numbers might be in order. Especially since the IMEI and IMSI numbers were reportedly part of the T-Mobile breach at least for some subset of the breached subscribers:

ICCID is the "**I**ntegrated **C**ircuit **C**ard **ID**". This is a unique and unchangeable international SIM card identifier. It's the 18 to 22-digit number that can be seen printed on the outside of the SIM and it smells like the work of a committee that got out of control. For example, every SIM ICCID begins with the digits 89. Why? Well, because this is an industry code that indicates that this is a product for use by telecommunications networks. But aren't all SIMs for use in telecommunications networks? Yes, of course... that's why the number is always 89. But, if it's always 89, then why have it at all? Exactly. Ask the committee. They're quite pleased with their work here.

Following the obligatory and entirely redundant 89 we have one to five digits for the country code. After the country code is the mobile network code (MNC), a string of one to four digits associated with the mobile network operator that issued the SIM card. This code represents the SIM card's home network. For example the MNC of 004 is the code for Verizon Wireless. The ICCID then ends with a guaranteed-unique string of digits which allows this SIM card to be uniquely identified everywhere and for all time. While it's possible to change the information contained within the SIM (including the IMSI), this identity of the SIM itself remains fixed at manufacture.

IMEI is the "**I**nternational **M**obile **E**quipment **I**dentity." It's a unique number also immutably assigned to every mobile handset or other cellular-capable device. And, as with the ICCID, this number is burned into each phone and cannot be changed.

IMSI this is "**I**nternational **M**obile **S**ubscriber **I**dentity" and as its name suggests, it is the thing that CAN be changed as a subscriber's phone number moves from device to device, and thus SIM to SIM. It's a unique identifier that identifies a subscriber to the wireless world. It specifies the country and mobile network to which the subscriber belongs.