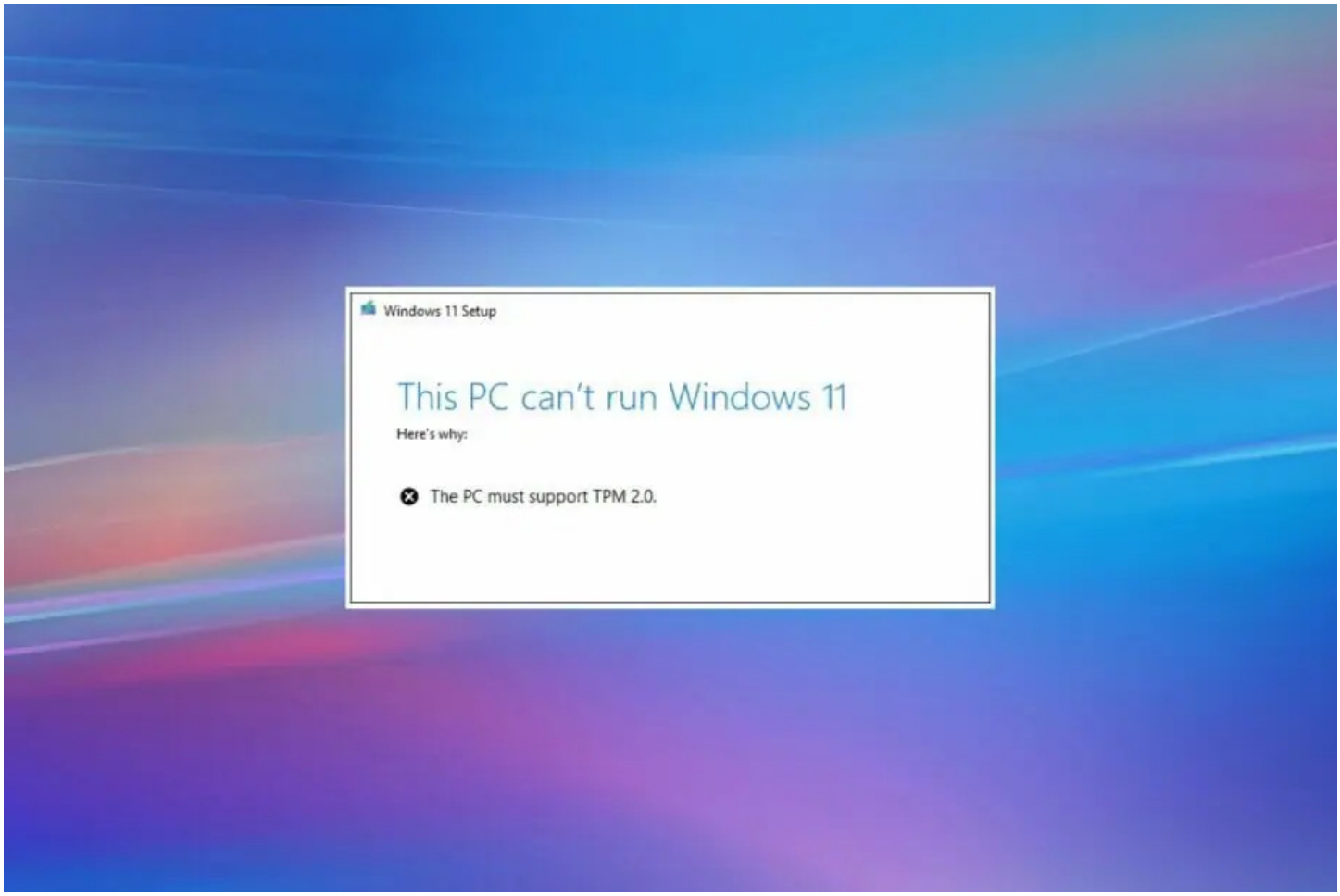
Security Now! #835 - 09-07-21

TPM v1.2 vs 2.0

This week on Security Now!

This week we look at a way of protecting ourselves from Razor-mouse-like local elevation of privilege attacks. We reexamine the meaning of the phrase “Internet Anonymity” following the ProtonMail revelation. We revisit Apple's now delayed CSAM plans. We look at some new troubles for BlueTooth and at a popular and persistently unpatched residential security system which can be trivially disarmed by bad guys. We share some interesting closing the loop feedback and a new Sci-Fi discovery. Then we take a long and careful look at the details and differences between version 1.2 and 2.0 of the Trusted Platform Module specification to discover just what it is that Microsoft wants to insist is available for Windows 11.



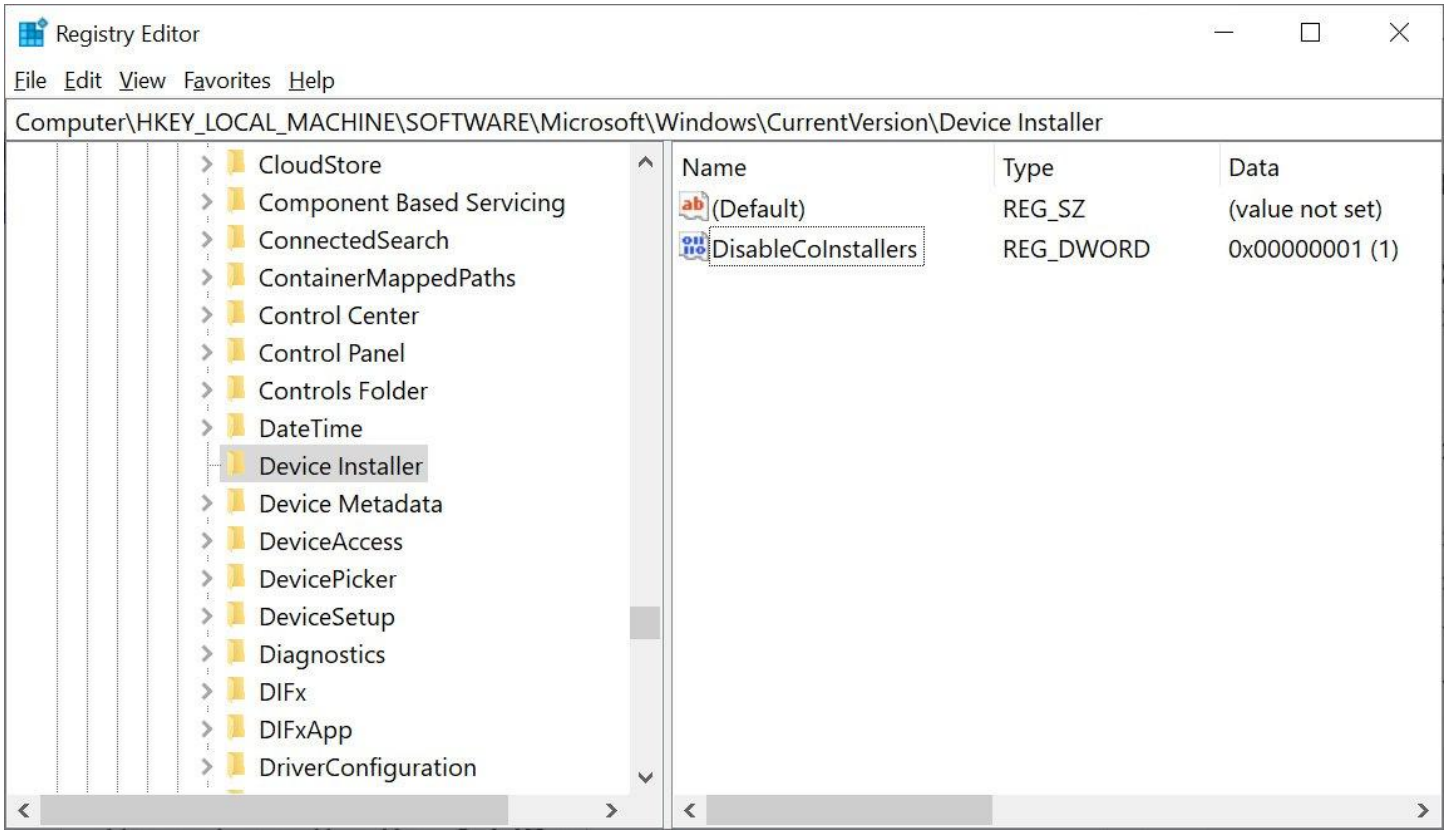
Security News

The Razor mouse & keyboard

Two weeks back we covered the industry's discovery that the tongue-in-cheek phrase "Plug And Pray" was more apropos than we imagined, with the news that plugging a Razor mouse or keyboard into any PC could be used to trivially and locally bypass that system's possibly carefully crafted user privilege restrictions to give the operator full system level rights. The worry was that since this was such a simple mistake to make it must certainly have also been made elsewhere. And, indeed, since that initial discovery additional similar instances are being discovered.

The good news is that Will Dormann at CERT's Coordination Center has found a registry key that governs Windows' ability to autonomously download and run any of those not-explicitly-asked-for-and-executed Plug and Play peripheral installation packages.
https://twitter.com/wdormann/status/1432703702079508480 Last Tuesday, Will Tweeted:

> Do you not like the fact that connecting a Razer device auto-runs an arbitrary installer w/ privs? Do you suspect that other devices may be exploitable? Fear not!
> Set HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Device Installer\DisableCoInstallers = 1



Thanks to BleepingComputer for their screenshot!

Now, I'm the guy who left the DNS IP address override in his ecommerce server's HOSTS file, causing its attempted connections to fail when our ecommerce provider's IP address changed, as it certainly had every right to. So, this suggests that if I were to make such a change, I might forget that I had done so and I'd be pulling my hair out wondering why something I had plugged in wasn't automatically installing its own software. I'm just sayin'.

For anyone — probably those with enterprise IT management responsibilities — who's interested, the details are in the show notes. And for anyone who is comfortable with the Windows Registry, there's nothing to it. You go to HKEY LOCAL MACHINE, then SOFTWARE, Microsoft, Windows, CurrentVersion, Device Installer. On my machine, that key existed but it was blank, without any assigned values. So, at that key you create a REG_DWORD value named "DisableCoInstallers" and set its value to "1". Presumably after a reboot, Plug And Pray will then become "Plug and go get the required installer yourself." Whereupon local rights would restrict the local user to downloading and running with the installer under their own privileges, if they're even able to do that.

Again, the danger is that someone will forget that this was done and be puzzled when things don't work as automagically as expected. But in many settings that might be exactly what's desired... so I wanted to be sure that our followers were aware of this option.

The wishful phrase "Internet Anonymity" is an oxymoron

Any longtime follower of this podcast would have seen example after example of the truth that while the Internet can make tracking people down much more difficult, it's still not impossible. This is why we've developed the model of having two people who wish to hold a private conversation, strip down to their shorts and walk out into the middle of Central Park with no one around carrying a large and thick quilted comforter blanket. They then throw the blanket over themselves, then whisper as quietly as they possibly can into each other's ear in order to exchange secrets.

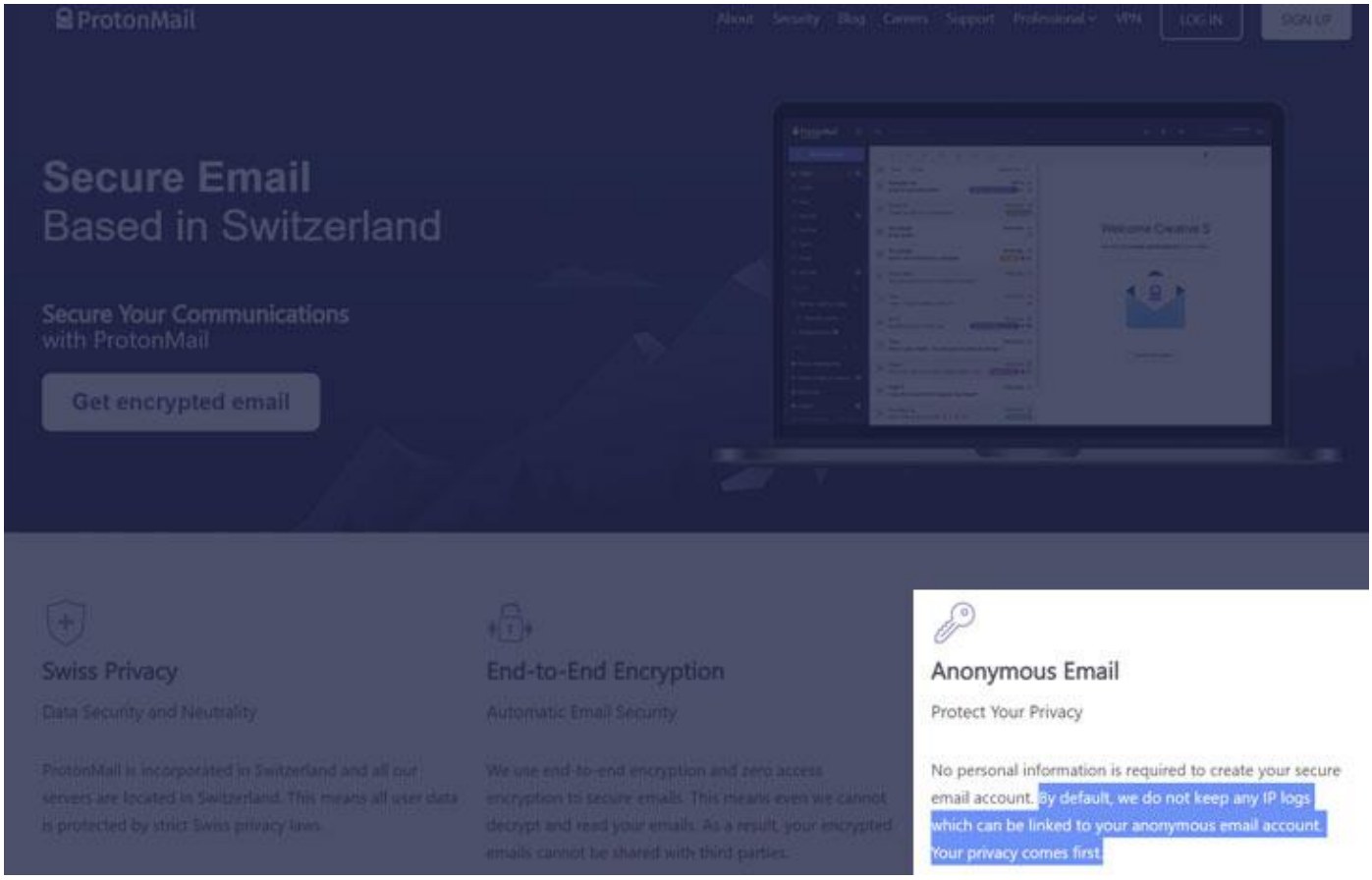
That's about as low-tech as anything could possibly be, which is exactly the point. Technology provides vast measures of convenience, but that convenience ALWAYS comes at the cost of true security.

In the doghouse this week is ProtonMail. ProtonMail boasts on its website "Secure Email Based in Switzerland" but that statement's meaning was changed last week. I just scrolled down through their homepage at ProtonMail.com and — oh boy! — it looks WONDERFUL!

The Hacker News writes that: *"On its website, ProtonMail advertises that: "No personal information is required to create your secure email account. By default, we do not keep any IP logs which can be linked to your anonymous email account. Your privacy comes first.""* But when I just went to that page and searched for bits of that assertion — it had apparently been removed. I wonder why?

Could it perhaps be, because it was recently discovered that ProtonMail had provided location information to Swiss authorities which directly led to the arrest of one or more of the users of their supposedly Swiss identity-protecting service? Yeah... that might have been a factor.

The Hacker News captured a "before" screenshot of ProtonMail's homepage. Right at the top it lists its main feature categories as: "Swiss Privacy", "End-to-end Encryption" and "Anonymous Email". But it doesn't say that today. Their homepage has been changed. Today, the top line still begins with "Swiss Privacy" and "End-to-end Encryption." But now the third item is "Your data, your rules." — where it previously said "Anonymous Email"



I give them serious props for fessing up and apparently thinking "Okay, uhhhhh... we really can't make that Anonymous Email claim anymore, can we?" That had to hurt!

ProtonMail acknowledged that it had received a "legally binding order from the Swiss Federal Department of Justice" related to a collective called Youth for Climate, which it was "obligated to comply with," compelling it to handover the IP address and all information it had, related to the type of device used by the group to access the ProtonMail account. Hmmmm. That's probably not the nature of the protection that those users of ProtonMail believed they were receiving after reading ProtonMail's original, quite powerful and compelling privacy expounding home page... which, as I've noted, no longer exists.

So, despite its no IP logs claims, the company acknowledged that while it's illegal for the company to comply with requests from non-Swiss law enforcement authorities, it will be required to do so if Swiss agencies agree to assist foreign services such as Europol in their investigations.

In part of a lengthy response posted on Reddit, the company said: "There was no possibility to appeal or fight this particular request because an act contrary to Swiss law did in fact take place (and this was also the final determination of the Federal Department of Justice which does a legal review of each case)."

Put simply, ProtonMail will not only have to comply with Swiss government orders, it will be forced to hand over relevant data when individuals use the service to engage in activities that are deemed illegal in the country. This includes monitoring its users IP addresses.
ProtonMail founder and CEO Andy Yen tweeted: "Proton must comply with Swiss law. As soon as

a crime is committed, privacy protections can be suspended and we're required by Swiss law to answer requests from Swiss authorities. It's deplorable that legal tools for serious crimes are being used in this way. But by law we must comply with Swiss criminal investigations. This is obviously not done by default, but only if legally forced."

Elsewhere the company bemoaned: "The prosecution in this case seems quite aggressive. Unfortunately, this is a pattern we have increasingly seen in recent years around the world (for example in France where terror laws are inappropriately used)."

It seems clear that their hearts are in the right place. They really and truly do want to be offering absolute privacy. But of course they are correct in explaining that IF they have the ability to comply with such requests, they must by law do so. What I wonder is whether it wouldn't be possible to design a system where it's just not possible for them to comply no matter how much they might be forced to?

For example, we've conjectured here that one possible motivation for Apple's quite unpopular plan to compare photo signatures against a known photo signature database before uploading new photos to iCloud, might actually be so that they can further lock down iCloud as they have their other end-to-end encrypted systems. In other words, where there's a will there's a way.

So, ProtonMail users who are concerned about the visibility of their IP addresses should also route their access through the Tor network to obtain additional anonymity. Since eMail is not real time communications anyway, routing eMail through Tor, which adds significant communications latency, makes a lot of sense.

And speaking of Apple's client-side image matching...

Apple has announced that it will be pausing the planned rollout of its quite controversial plan to screen users' devices for child sexual abuse material (CSAM). As we know, Apple received substantial and sustained pushback, not only over worries that the technology could be weaponized for mass surveillance and erode the privacy of users, but just over the "ick factor" at the idea of having anything to do with CSAM materials pre-loaded into all iOS devices which are capable of upload images iCloud.

Apple wrote: "Based on feedback from customers, advocacy groups, researchers, and others, we have decided to take additional time over the coming months to collect input and make improvements before releasing these critically important child safety features."

In this instance, I give Apple credit for publishing their intentions for this technology well ahead of its planned incorporation into iOS 15, and in listening to and responding to all of the blowback they received. The trouble is, from a technology standpoint, it's pretty much binary. Either you do it this way or you don't. But, if their underlying motivation was to somehow arrange to lock down iCloud in an end-to-end encrypted fashion, they might be able to do that by offloading the image analysis to an intermediate cloud-based service which would then encrypt the image on behalf of the image's iOS user. So, this might mean that they'll just need to work a bit harder and that everyone will have learned a valuable lesson. Everyone wants this to be done, just not on their own handsets.

BlueTooth has new troubles

The new set of attacks are known as "BrakTooth" ("Brak" being Norwegian for "crash"). Paraphrasing from the discoverers' disclosure, they explain:

> *Bluetooth Classic (BT) protocol is a widely used wireless protocol in laptops, handheld devices, and audio devices. In the past few years, Bluetooth has come under scrutiny due to the discovery of several critical vulnerabilities. In this report, we disclose BrakTooth, a family of new security vulnerabilities in commercial BT stacks that range from denial of service (DoS) via firmware crashes and deadlocks in commodity hardware to arbitrary code execution (ACE) in certain IoTs. As of today, we have evaluated 13 BT devices from 11 vendors. We have discovered a total of 16 new security vulnerabilities, with 20 common vulnerability exposures (CVEs) already assigned and four (4) vulnerabilities are pending CVE assignment from Intel and Qualcomm.*
>
> *All the vulnerabilities are already reported to the respective vendors, with several vulnerabilities already patched and the rest being in the process of replication and patching. Moreover, four of the BrakTooth vulnerabilities have received bug bounty from Espressif System and Xiaomi. An exploration of Bluetooth listings reveals that BrakTooth affects over 1400 product listings. BrakTooth exposes fundamental attack vectors in the closed BT stack.*
>
> *As the BT stack is often shared across many products, it is highly probable that many other products (beyond the ˜1400 entries observed in Bluetooth listing) are affected by BrakTooth. Therefore, we suggest vendors producing BT system-on-chips (SoCs), BT modules or BT end products to use the BrakTooth proof-of-concept (PoC) code to validate their BT stack implementation. The availability of the BrakTooth PoC is discussed at the end of this report.*

Here again we have a problem with the way we're still doing technology — like with the idea that we're going to use and trust proprietary voting machines. In both the BlueTooth and the voting machine cases, a closed system must be trusted to do the right thing while its NOT doing the right thing can have sweeping consequences for millions of people. Yet we just shrug. "Oh darn."

We already know that opening everything up doesn't, by itself, automatically make everything secure. So while it's not sufficient, it is almost necessary. To my mind, this is the single most compelling reason to push for opening our technologies.

Here again, with BrakTooth, we have researchers who have been forced to hack into and reverse engineer important implementations of technology that will be used by millions of people and once shipped and sold will almost certainly never be updated in the field. The good news — this time — is that being BlueTooth the attack range is short. And that most of the flaws found simply crash (or "Brak") the stack. So, whew. But this is another of those things where there are doubtless many agencies spread around the world who are sucking up these vulnerabilities and may very well have applications for many of them today. How many movies have we seen where someone "runs a bypass" on a keypad to gain entry to a protected area?

That fictional scenario is becoming less and less far fetched with each passing year — or month. The trouble is, these IoT things will probably never be fixed. Microsoft cannot manage to get their deployed Exchange Servers patched. So what chance does the manufacturer of a residential alarm system have? It's got to be near zero. But first they have to want to, and care to, for there to be any chance at all. But why would they? They've already got their customer's money.

We are rapidly filling the world with a bunch of incredibly complex crap that ships the moment it stops crashing in the lab. And that's an entirely different result from having potentially mission-critical technologies being actively resistant to attack and abuse. These researchers just showed us once again that the world we're living in today is one where hoping for the best is apparently all we can do.

Attackers Can Remotely Disable Fortress Wi-Fi Home Security Alarms

And speak of the devil, here's a perfect example:

Two vulnerabilities have been discovered in the Fortress S03 Wi-Fi Home Security System that could be abused by a malicious party to gain unauthorized access with an aim of altering system's behavior, including disarming the devices without the victim's knowledge.

https://www.amazon.com/stores/Fortress+Security+Store/page/C2C11927-72E1-47FD-BB3F-7FED53E608E8

The two unpatched issues were discovered and reported by Rapid7 back in May and they honored a 60-day disclosure deadline. So let me say that again: Fortress was notified of these problems back in May. They've nothing.

The Fortress S03 Wi-Fi Home Security System is a do-it-yourself (DIY) alarm system that enables users to secure their homes and small businesses from burglars, fires, gas leaks, and water leaks by leveraging Wi-Fi and RFID technology for keyless entry. According to Fortess' website, it's security and surveillance systems are used by "thousands of clients and continued customers."

Rapid7 described the vulnerabilities as being "trivially easy to exploit," noting that CVE-2021-39276 concerns an unauthenticated API Access that enables an attacker in possession of a victim's email address to query the API to leak the device's International Mobile Equipment Identity (IMEI) number, which also doubles up as the serial number. Armed with the device's IMEI number and the email address, the adversary can proceed to make a number of unauthorized changes, such as disabling the alarm system via an unauthenticated POST request without the owner's knowledge. While this is not usually much of a concern for random, opportunistic home invaders, this is particularly concerning when the attacker already knows the victim, such as an ex-spouse or other estranged relationship partner.

The second vulnerability (CVE-2021-39277) presents similar problems but requires less prior knowledge of the victim, as the attacker can simply stake out the property and wait for the victim to use the RF-controlled devices within radio range. The attacker can then replay the "disarm" command later, without the victim's knowledge.

And, now that Rapid7 has made their disclosure — which of course includes full details — anyone with a bit of tech savvy can have at it since Fortess apparently isn't concerned and/or may not be able to field update their devices even if they were.

Once again, the world we're living in today.

Closing the Loop

Steven / @wnevets
I don't know if anyone has suggested this but I use Google voice to secure my SMS fallbacks for services that require it. Taking over my Google Voice number requires taking over my Google account which doesn't have SMS fallback enabled

Jon @JonMcD86
Hey Steve, In this week's Security Now episode you touched on the perils of using SMS messaging to your cell phone for authentication. I just wanted to throw one thing out there that I do that others might find useful. For platforms that require SMS authentication I like to use a Google Voice phone number to receive text messages that I protect using my personal Yubikey. It's annoying having to sign into this account repeatedly and it's far from a perfect solution, but I feel like it's better than leaving this, as you put it, backdoor into my services where SMS recovery is a requirement.

Anyway, huge fan and I listen every week. Keep up the great work.

Ant Lednev / @AntLednev
Hi Steve, as I am watching SecuritNow and where you describe the TMobile, SMS, and LastPass with SMS issues (as well as banking requiring SMS) - just want to mention that I am using Google Voice for this purpose. It is protected by Google login and 2FA, has an iPhone app, and you can access it from the browser too. Curious what are your thoughts on replacing cell SMS to VoIP SMS services.

Alain / @Alain_Gyger
I've really come to like @Tailscale as my VPN replacement ... enough, in fact, to write a review about it (https://www.thegygers.com/2021/09/03/tailscale-review/) Thanks @SGgrc for mentioning it on @SecurityNow

First off … I've tried configuring WireGuard multiple times (around 6 by my count) on different devices (an Ubuntu VM, a few Raspberry PIs, and a pfSense firewall) all with various versions of success. None of my attempts ended up working fully though. I would be able to access internal resources but not the internet, the internet but nothing internal, or nothing at all (by this point there would be a Raspberry Pi shaped hole in the window).

So I gave up (bad tech guy, go sit in the corner and use a Windows machine for a while).

Then, during my weekly listen of SecurityNow, Steve Gibson mentioned Tailscale as a service that uses WireGuard (episode 830) and that it's really easy to configure. I figured, why not … let's give it another go … and I like it!

The highlights:

Works on Ubuntu (or any Linux distro for that matter)
Works on Mac/iOS devices
Works on Windows (yay … I guess?)
Works on Android
It's fast
The free tier is very generous (the paid versions aren't bad either, especially "Subnet router failover" on the Business plan!!)

Walt Stoneburner / @waltomatic
Steve, enjoyed your last show's quick dive on assembly language and shout out to SoftICE (oh, how I miss using that). While I hope to high heaven that someday you write an assembly language book that goes into your own personal thoughts and techniques based on years of real world usage, are there any resources you'd recommend over any others for experienced C programmers?

I replied to Walt, recommending that if he's able to find anything written by Michael Abrash on eBay or in any used book store online or offline, that he'd never regret spending a few bucks on anything that Michael Abrash ever took the time to write. I own:

- The Zen of Assembly Language, Volume 1: Knowledge.
- The Zen of Code Optimization
- The Zen of Graphics Programming, and
- Michael Abrash's Graphics Programming Black Book (a collection of his various articles)

They were written back in the days when memory and CPU cycles were both ultra scarce and thus highly prized. Sadly, their time has passed. I'm still writing that way myself, not because it makes any practical sense, but because I love moving known-size data around between registers and memory. The best analogy might be to a craftsman who loves to build homes by hand. Loves measuring each piece of timber, plane-ing its surface and fitting the perfectly sized pieces together. Pounding each nail in squarely. And once that's done, standing back, taking a slow deep breath and taking quiet personal pride in a really nice piece of work. And it doesn't matter whether anyone else appreciates it or even ever sees it. It wasn't done for anyone else. It's something you did for yourself. For the sheer joy of using your hands to solve a collection of little puzzles. I've never found anything as satisfying as coding.

Anyone who really loves coding, for its own sake, in any language, might Google the phrase "Programming Gems." That's sort of become the hook phrase for the art form side of coding. Coding and algorithms are complex enough that particularly elegant solutions often exist. I recall that back at the start of my implementation of SQRL, I needed to quickly look up a user-interface string by its index. So I needed a dictionary that I could access at very high

speed. The way to do that is with a binary search and though I've implemented countless binary searches through the years, the planets must have been in perfect alignment that day because I wrote the most elegant binary search for 32-bit Intel assembler that I'd ever written in my life. It's use of registers, condition codes and conditional branching was sublime. And yeah... I know... talk about hyper-geeky. But that's where I live.

Steve YATES / @sayates
Thanks @SGgrc , #Initdisk has just rescued a 64GB USB key that failed to respond to anything else.

Sci-Fi

Believe it or not, Peter Hamilton has been involved in the creation of a shorter book. The title is "Light Chaser" by Peter F. Hamilton and Gareth L. Powell. It was quietly released two weeks ago, on August 24th. And I wouldn't have been aware of it if I hadn't been looking for something else when Amazon brought it to my attention.

The Nerd Daily:
https://thenerddaily.com/review-light-chaser-by-peter-f-hamilton-gareth-l-powell/

Peter F. Hamilton and Gareth L. Powell team up in this explosive, action-packed novella about a love that transcends lifetimes and is powerful enough to destroy an empire.

Amahle is a traveller who sails through the universe with nothing for company but the ship’s AI. Known throughout the universe as a Light Chaser, her route takes her to worlds throughout The Domain, where she collects memories in exchange for various goods.

But when she discovers memories from different lives and different worlds that are meant for her, and seem to be from the same person, she begins to question her entire existence. Each memory unlocks Amahle’s own memories and slowly reveals an elusive enemy with a horrifying plan. Amahle realises she’s the only one who can do anything to stop it, but it will cost her everything.

For a shorter book, this story packs an incredible punch. Light Chaser comes in at 173 pages, yet it is epic and expansive, taking us through worlds and lifetimes in rapid succession. Rather than feeling rushed or lacking, the prose is razor sharp, carving out exactly what we need to understand the world and the technology while propelling us forward.

TPM v1.2 vs 2.0

Let's begin by discussing why any of this TPM stuff is needed at all.

The presumption is that once an operating system has taken possession of a system's hardware, configured and placed its processors into their protected modes, there's no possible way for anything to happen on the system that the operating system cannot intercept and consider. In other words, programs running under such an operating system are 100% absolutely contained as an OS client. They are unable to do anything that the operating system does not explicitly allow. Any unallowed action, such as attempting to touch the system's underlying hardware or read or write outside of fixed boundaries will raise an exception condition. This causes the offending client instruction to be suspended before it's able to take its action and returns control to the operating system for consideration of what to do. Now, this might be something benign, such as a client attempting to access some valid memory that it does have access to, but which has been swapped out of RAM to the system's virtual memory backing store. In such a case the OS will schedule the RAM to be retrieved for its client and will briefly suspend the client until its instruction that triggered the memory exception can be restarted to then succeed.

In an ideal world, once it's started, the operating system has all the tools it needs to actively supervise, manage and control all of the subsequent activity occurring on the system. And for the most part this works. As we know it's still an imperfect system because operating systems have become incredibly complex and human programmers make mistakes. As a result, the designers of malicious software keep finding ways to circumvent this control to elevate the privileges of the clever clients they design. This allows their malicious clients to obtain the same access rights and privileges as the underlying operating system, thus completely escaping the operating system's control to obtain free reign over the system.

So the ambition of obtaining truly secure computing continues to face the challenge of maintaining total control over its deliberately misbehaving malicious clients once it is in control. But this assumes that the operating system has not somehow been maliciously modified either while it's in cold storage or before it's able to finish booting to then assume control over the systems' processors and other hardware. In other words, it is becoming quite difficult to attack an operating system once it has control of the system because, mistakes notwithstanding human error, modern processor technology can provide absolute control. But what protects the operating system itself from being maliciously modified before it has the opportunity to begin protecting itself?

The answer is the Trusted Platform Module, or more generically, some form of Hardware Root of Trust (HRoT). This is shortened, popularized and simply referred to as "Secure Boot." Security Now! Episode 500, which we recorded a little over six years ago on March 24th, 2015, was titled "Windows Secure Boot." So I'll refer any of our listeners who want more details about exactly how this works step-by-step to that podcast which remains entirely relevant today.

The simple fact is, secrets are required for security. And the protocols which are used to verify secrets must be tamperproof.

There must be some things that an attacker does not know, cannot know, and cannot manipulate. This will prevent an attacker from subverting the operation of an otherwise completely open and publicly known system.

We're only able to believe that we're connecting to the proper remote Internet service because the certificate it presents contains a valid signature that has been signed by the secret private key maintained by a 3rd party whom we trust to have verified the attested identity of the remote service.

When two agencies wish to conduct a private conversation in plain sight, they use a key agreement protocol to derive a temporary secret key which they'll then use to encrypt their subsequent communications.

No matter how open any system is, there must be some place where secrets can be privately manipulated so that no one -- not even someone having complete access and visibility into the system -- can interfere with the system's intended operation. In the case of a personal computer, where an attacker, or an attacker's code, might have access to the system before the operating system has booted, there must be a little black box of some kind which is beyond any attacker's ability to interfere or manipulate.

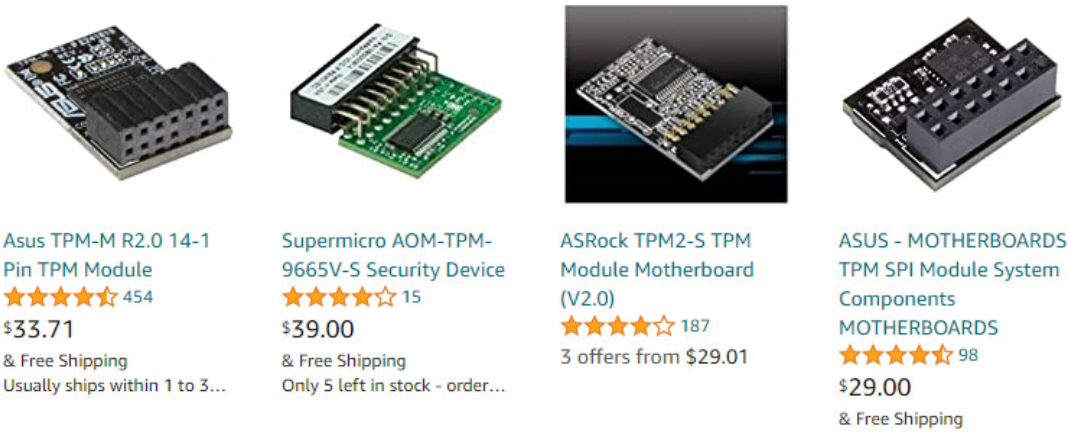
The Trusted Computing Group has carefully defined a minimal set of features and functions to standardize the operation of such a black box and has named it the Trusted Platform Module.

As suggested by the title of this podcast, and of the controversy surrounding Microsoft's determination to raise the version requirements of a system's TPM from v1.2 to v2.0, there have been and are two major release versions of the trusted platform module.

TPM 1.2 was first released 16 years ago in 2005 and it received its final revision in 2011.
TPM 2.0 was first released 9 years ago in 2014 and was last tweaked two years ago in 2019.

The reason it’s possible to have a first release of a v1.2 or 2.0 with subsequent tweaks that don’t change the version is that the 1.2 and 2.0 refer to API definitions rather than implementations.

The TPM is often built into a system as a tiny stand-alone chip soldered onto a laptop or desktop motherboard. The Gigabyte desktop motherboard I'm using has a socket for an optional TPM and, as we can see from the Amazon screenshot below, this is a common option:



The historical logic employed by motherboard makers was that even though a TPM has been available for the past 16 years, and it's possible that someone might want one, no one really needed it or cared. So why burden the cost of their motherboards with something that would sit there unused. As we know, this thinking will now be changing with a vengeance. Note that since a TPM is actually a set of callable functions, there's nothing preventing it from being incorporated into the system's core silicon chipset. And that's exactly what Intel has done. Ever since Skylake (6th gen), nearly all Intel CPUs have an embedded TPM 2.0 that Intel calls Platform Trust Technology (PTT). AMD CPUs have an embedded TPM 2.0 called fTPM from 2016's AM4 platform. It's also possible to do the entire TPM in software -- Microsoft has a pure software-only simulator -- but doing so loses all of the true security advantages of having secrets protected within an impenetrable black box.

So what is TPM? The Trusted Computing Group's own white page describing TPM is dry, incomplete, and states that TPM v1.2 is the latest current version. So someone hasn't been paying much attention to keeping it up to date. To begin to answer this question we'll turn to the source of all the controversy, Microsoft. They both helped design and, of course, use the TPM. What they wrote might be a bit self-aggrandizing, but I want to share it since it provides a few surprising bits and pieces that aren't found elsewhere...

Microsoft has led the architecture and adoption of the TPM since its inception. Microsoft invented and contributed the attestation, sealing and Platform Configuration Register (PCR) features to the original TPM, and contributed to the overall design.

More recently, Microsoft architected and edited the TPM2.0 specification. Many new concepts and features were introduced with TPM2.0, including crypto-agility, easier management, a more flexible authorization model, and better extensibility. TPM2.0 devices are now available from many vendors, and are incorporated into most business class PCs and many servers. TPM2.0 is also making increasing inroads into network equipment, mobile and IoT devices.

The TPM2.0 specification is unique in that it is machine readable. Most of the normative behavioral specification is written in a subset of the C programming language, and the TPM programming interface is defined in machine-readable tables. This allows vendors to quickly build high-quality and interoperable TPM implementations.

The TPM is a low-cost, but powerful and flexible, crypto-processor. A TPM does many of the things that a smart-card or hardware security module (HSM) does – for example, it is able to create, manage and use cryptographic keys, as well as store confidential data. But a TPM is intimately tied into how a computer boots and runs, which means it is far more powerful and useful than a simple "smart-card on the motherboard."

For example, platforms that incorporate TPMs "measure" and log the software that boots on the device. The resulting boot-log can be used to verify that devices are running known-software and are up-to-date using a TPM feature called quoting or attestation. The boot-log can also be used to protect keys for disk encryption, because the TPM incorporates a feature called sealing that can be used to make sure that the encryption key is only disclosed to authorized software, and not to disk-cracking tools.

Other advanced TPM features include a secure clock, monotonic counters, a non-volatile storage facility, and very flexible and secure mechanisms for key management operations like key import and export.

Okay, so we have an overview.

Most of the articles describing TPM 1.2 vs 2.0 get stuck on the fact that 2.0, being newer, adds support for some additional newer and stronger cryptographic primitives. Specifically, the original TPM 1.2 builds-in functions for SHA-1, HMAC-160, 1024 and 2048-bit RSA public key ciphers. To these TPM 2.0 adds 256-bit flavors of SHA and HMAC, so SHA2 256 and HMAC-256. 2.0 also adds several 256-bit elliptic curve ciphers and elliptic curve Diffie-Helman. Since TPM 2.0 is backward compatible with TPM 1.2, software could simply continue using the TPM 1.2 functions to run on platforms having either TPM standard. And this is, of course, what Windows 10 does today and will continue to do for the next four years until October 14th, 2025 which is the targeted end-of-support for Win10.

But Microsoft may have decided that it's time to force a change. We've seen and discussed countless examples of this through the life of this podcast. It's no longer possible to connect to remote web servers over SSL 2 and usually not using SSL 3. And when I go to Ivan Ristic's terrific SSL Labs facility to ask it about GRC.COM it says: "This server supports TLS 1.0 and TLS 1.1. Grade capped to B." So you can't support anything older than TLS v1.2 if you want to get an "A" from Ivan.

And we know that forcing a change in protocols — just for the sake of forcing a change when the currently used protocols appear to be working just fine — is always going to be painful. So if Microsoft wants to force their Secure Boot and other TPM dependent technologies forward with Windows 11, they'll want to drop support for the older SHA-1, and HMAC-160 and insist upon having Windows 11 only rely upon and use the newer 256-bit crypto technologies.

But there are three things that Microsoft may have missed:

- The first is that dropping old and adding newer crypto technologies has traditionally been a matter of upgrading a server or client's software. If your browser doesn't support the latest protocol, click "upgrade." But in the case of TPM, the cryptography is locked into the hardware. By design, it cannot be upgraded because it must be designed to be an inviolate black box. So while it's easy for Microsoft to say "We're going to require the use of 256-bit crypto technologies which are missing from TPM 1.2 and are present only in TPM 2.0", not only is it not easy, it's not possible for Windows users to fix this with a software upgrade.

- The second thing they've missed is the lack of any compelling reason to move from Windows 10 to 11. During last week's Windows Weekly MaryJo expressed her puzzlement about the entire thing, asking Paul whether there was anything truly new about Windows 11 beyond — and she really asked this — its centered Start menu and its rounded corners. This was MaryJo Foley who would know, if anyone in the world would, right? She would normally be all over it. Paul shrugged and joked that there were, after all, **quite a few** corners that had needed to be rounded during the creation of Windows 11.

- And the third thing Microsoft may have missed is that many people using Windows 10 today are not using ANY features of TPM — regardless of edition. They aren't booting with secure boot and they don't use Bitlocker. They don’t want them or care about them. So the security profile of these people suggests that TPM doesn't matter at all to them. Yet Microsoft is planning on denying these people the option of moving to Windows 11 for no reason other than that features they are not using, and presumably have no interest in ever using, are absent from their hardware. *“No nicely rounded corners for you!”*

So, Microsoft appears to be saying: We've decided to force a change to Windows, that no one really needs, because we want to force people to use more modern hardware even if they don’t need more modern hardware. This doesn't feel like a winning strategy to me.

When I was introducing TPM 2.0 I said that most of the articles describing TPM 1.2 vs 2.0 get stuck on the fact that 2.0, being newer, adds support for some additional newer and stronger cryptographic primitives. The point I was making is that there's more to 2.0 than stronger and newer crypto.

First, I'll enumerate the five major functions that TPM 1.2 offers, then the six additional features which the architects of TPM added in the move from 1.2 to 2.0. So, TPM 1.2 provides for:

• Identification of devices: Prior to the release of the TPM specification, devices were mostly identified by MAC addresses or IP addresses—not security identifiers.

• Secure generation of keys: Having a hardware random-number generator is a big advantage when creating keys. A number of security solutions have been broken due to poor key generation.

• Secure storage of keys: Keeping good keys secure, particularly from software attacks, is a big advantage that the TPM design brings to a device.

• NVRAM storage: When an IT organization acquires a new device, it often wipes the hard disk and rewrites the disk with the organization’s standard load. Having NVRAM allows a TPM to maintain a certificate store.

• Device health attestation: Prior to systems having TPMs, IT organizations used software to attest to system health. But if a system was compromised, it might report it was healthy, even when it wasn’t.

With the passage of time and with the benefit of experience with TPM 1.2, TPM 2.0 then added…

• Algorithm agility: Algorithms can be changed without revisiting the specification, should they prove to be cryptographically weaker than expected.

• Enhanced authorization: This new capability unifies the way all entities in a TPM are authorized, while extending the TPM’s ability to enable authorization policies that allow for multifactor and multiuser authentication. Additional management functions are also included.

• Quick key loading: Loading keys into a TPM used to take a relatively long time. They now can be loaded quickly, using symmetric rather than asymmetric encryption.

• Non-brittle PCRs (Platform Configuration Registers): In the past, locking keys to device states caused management problems. Often, when a device state had to go through an authorized state change, keys had to be changed as well. This is no longer the case.

• Flexible management: Different kinds of authorization can be separated, allowing for much more flexible management of TPM resources.

• Identifying resources by name: Indirect references in the TPM 1.2 design led to security challenges. Those have been fixed by using cryptographically secure names for all TPM resources.

For more on the details of TPM 2.0, I'll refer our podcast followers to a terrific free guide titled: "A Practical Guide to TPM 2.0" published by Springer Link
https://link.springer.com/content/pdf/10.1007%2F978-1-4302-6584-9.pdf
https://link.springer.com/download/epub/10.1007%2F978-1-4302-6584-9.epub

Microsoft's chart of their technologies which require TPM 2.0 and won't run under TPM 1.2 is available here: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/tpm-recommendations#tpm-and-windows-features

	TPM 1.2	TPM 2.0
Measured Boot	✅	✅
BitLocker	✅	✅
Device Encryption	❌	✅
Windows Defender Application Control	✅	✅
Windows Defender System Guard	❌	✅
Credential Guard	✅	✅
Device Health Attestation	✅	✅
Windows Hello	✅	✅
UEFI Secure Boot	✅	✅
TPM Platform Crypto Provider Key Storage Provider	✅	✅
Virtual Smart Card	✅	✅
Certificate storage	✅	✅
Autopilot	❌	✅
SecureBIO	❌	✅

Measured Boot instruments the booting process, storing history in the TPM to detect future tampering. Microsoft says that it runs fine under TPM 1.2 right now.

Microsoft says that BitLocker runs equally well under TPM 1.2 and 2.0.

Microsoft says that "Device Encryption" requires TPM 2.0 and that it's not available with TPM 1.2. But since Bitlocker, which is superior anyway, runs with either of the TPM's, unless you're a Windows Home user without Bitlocker, the lack of Device Encryption is a "who care?"

Windows Defender Application Control runs under either TPM. Microsoft explains:

"For Microsoft Edge, Application Guard helps to isolate enterprise-defined untrusted sites, protecting your company while your employees browse the Internet. As an enterprise administrator, you define what is among trusted web sites, cloud resources, and internal networks. Everything **not** on your list is considered untrusted. If an employee goes to an untrusted site through Microsoft Edge, the site is opened in an isolated Hyper-V-enabled container." — And that works fine under either TPM.

Windows Defender System Guard is a post-boot integrity verifier which works with the TPM to verify that Windows wasn't compromised during the boot. For that it needs TPM 2.0.

Seven other security features: Credential Guard, Device Health Attestation, Windows Hello, UEFI Secure Boot, TPM Platform Crypto Provider Key Storage Provider, Virtual Smart Card and Certificate Storage All work fine with either TPM 1.2 or 2.0.

"Windows Autopilot" does require TPM 2.0. I had to look up that one. Microsoft describes it as a collection of technologies used to set up and pre-configure new devices, getting them ready for productive use. Windows Autopilot can be used to deploy Windows PCs or HoloLens 2 devices. So no big loss there if we don't have TPM 2.0.

And, finally, SecureBIO also known as "Microsoft Enhanced Sign-in" needs TPM 2.0. This is an enterprise-targeted biometric identity system, even though Windows Hello works just fine with only TPM 1.2.

So, in conclusion, there are no obvious show stopping technologies present in TPM 2.0 that we cannot readily live without if all we have is TPM 1.2. If all we have is TPM 1.2, we get BitLocker but not the watered down feature-stripped Device Encryption. We don't get the enterprise double-check after boot System Guard, but no one running Windows outside of an enterprise would ever have it anyway. We also don't get Autopilot or SecureBIO, even though we do get fully functional Windows Hello.

In other words, while TPM 2.0 might enable a few extra features for the enterprise, TPM 1.2 provides everything that typical Windows users need. Which is what we have today.

And now, finally, here's the big "mic drop" moment:

Everything I've just enumerated ALL refers to the way everything is TODAY under Windows 10 — NOTHING CHANGES under Windows 11!

In other words, there are **NO NEW FEATURES** being brought to Windows by Windows 11 that create **ANY additional need** or reason for TPM 2.0.

When MaryJo asked Paul what was new in Windows 11 aside from the centered Start menu and the rounded corners, she was really asking. There are NO NEW FEATURES in Windows 11 that require anything more of the TPM than Windows 10 already does... yet Windows 11 is refusing to run on the same TPM's as Windows 10... apparently because someone at Microsoft thought it would be cool to enact a more restrictive change in requirements.

Given these realities, the path Microsoft should take for Windows 11 is clear: Simply use the maximum security that's being offered by whatever, if any, TPM is present in a system. If the platform offers TPM 2.0, great! Use the 256-bit enhanced security that's available there. If not, settle for the 160-bit security offered by SHA-1 and TPM 1.2's HMAC — just as Windows 10 does now. If a platform doesn't offer TPM 2.0, then its user cannot take advantage of those **four** enterprise-oriented features from among the **fourteen** that will run on **any** TPM.

Fine. So, explain to those enterprise users that if they want those **four** features they'll need to upgrade their hardware. But don't tell any random home or small business user, who couldn't care less about Windows Defender System Guard and Autopilot, that they're S.O.L. if they wish to upgrade to the new Windows ... just because it's possible to be mean. That's how it's going to be seen. It's going to be seen as capricious and arbitrary, because as we've just seen, it is.

It's my sincere hope that this decision is being given a broader airing within Microsoft, and that it's a requirement that they might reconsider. Given a careful analysis of the two TPM feature sets, and of Windows current use of them, there doesn't appear to be any compelling need to force a move to TPM 2.0 "just because."

P.S.: Windows 11 Dark Mode will have more soothing sounds.